A Comparative Study of Data Transformations for Efficient XML and JSON Data Compression

An In-Depth Analysis of Data Transformation Techniques, including Tag and Capital Conversions, Character and Word N-Gram Transformations, and Domain-Specific Data Transforms using SMILES Data as a Case Study

Shagufta Anjum SCANLON

A thesis submitted for the degree of
Doctor of Philosophy

School of Electrical Engineering and Computer Science
University of Bradford
2015

Shagufta A. Scanlon

A Comparative Study of Data Transformations for Efficient XML and JSON Data Compression

An In-Depth Analysis of Data Transformation Techniques, including Tag and Capital Conversions, Character and Word N-Gram Transformations, and Domain-Specific Data Transforms using SMILES Data as a Case Study

Keywords: XML, JSON, Compression, Decompression, Transformation, N-Grams, SMILES

Abstract

XML is a widely used data exchange format. The verbose nature of XML leads to the requirement to efficiently store and process this type of data using compression. Various general-purpose transforms and compression techniques exist that can be used to transform and compress XML data. More compact alternatives to XML data have been developed, namely JSON due to the verbosity of XML data.

Similarly, there is a requirement to efficiently store and process SMILES data used in Chemoinformatics. General-purpose transforms and compressors can be used to compress this type of data to a certain extent, however, these techniques are not specific to SMILES data.

The primary contribution of this research is to provide developers that use XML, JSON or SMILES data, with key knowledge of the best transformation techniques to use with certain types of data, and which compression techniques would provide the best compressed output size and processing times, depending on their requirements.

The main study in this thesis, investigates the extent of which using data transforms prior to data compression can further improve the compression of XML and JSON data. It provides a comparative analysis of applying a variety of data transform and data transform variations, to a number of different types of XML and JSON equivalent datasets of various sizes, and applying different general-purpose compression techniques over the transformed data.

A case study is also conducted, to investigate data transforms prior to compression to improve the compression of data within a data-specific domain.

Declaration

The candidate confirms that the work submitted is her own and that appropriate credit has been given where reference has been made to the work of others.

Shagufta Scanlon

Acknowledgements

I would like to thank both of my supervisors, Mr Mick Ridley and Dr Andrea Cullen, for their continued support in my research and professional development over the years, and for their advice and guidance throughout various stages of my research.

I would also like to thank my partner for his support, understanding and patience throughout my PhD research.

Publications

Conference proceedings

Scanlon, S., Ridley, M.: A Fully Reversible Data Transform Technique Enhancing Data Compression of SMILES Data. In: Proceedings of the IFIP WG 8.4, 8.9, TC 5 International Cross-Domain Availability, Reliability, and Security Conference in Information Systems and HCI (CD-ARES 2013), Regensburg, Bavaria, Germany, September 02-06, pp. 54-68 (2013)

Table of Contents

List of Figures

Appendix D: XML and JSON Overall Result Graphs and Tables

Appendix E: SMILES Overall Result Graphs and Tables

List of Tables

Appendix D: XML and JSON Overall Result Graphs and Tables

Appendix E: SMILES Overall Result Graphs and Tables

Chapter 1

Introduction

This thesis is concerned with enhancing the data compression of common data exchange formats, namely XML and JSON, and data within a specific domain, such as the SMILES chemical linear notation used in Toxicology, by applying compression over data transforms and transform variations.

1.1 Context of the Study

Extensible Markup Language (XML) was developed as a standard information exchange format across the World Wide Web (WWW) [32], [13], [46], [6], [43]. Prior to XML, data exchange between applications was difficult due to the application-specific nature of data storage and representation amongst applications. Thereby, the development of the standard XML format, not tied to a specific application, simplified information exchange between applications [78], [13], [46].

XML is considered a meta-language that contains self-describing properties to describe the nature of the XML data contained within the document. User-defined tags within the XML document structure are used for marking up the data, this is also referred to as the markup language, thus providing meaningful tags to describe the document [71], [70]. Whilst the self-describing nature of the structure of an XML document allows for flexibility, the verbose nature of these repeating tags negatively impacts data storage and processing times [71], [70], [76], [46].

1.2 Problem Statement and Study Motivation

The volubility of XML is due to the repetitive tags in the XML document structure present for each instance of an element [35]. Therefore, on a structural level, the verbosity of the XML data representation leads to high levels of redundancy [7], [9], [71], [70], [53], [35], [4], [13], [46].

The verbose nature of XML documents results in large document sizes which impacts data storage, processing and exchange costs [53], particularly for XML documents that contain deeply nested tags, and for those that include long tag and attribute names [71], [70]. The vast increase in size imposed by the XML document structure impacts the practical usage of the data in XML documents [13], [46].

The verbose nature of XML leads to the requirement to efficiently store and process this type of data using data compression. Various general-purpose compression techniques exist that can be used to compress XML data, however, these are text-specific and are not tailored specifically for XML data. Although XML-specific data compression techniques do exist that take into account the structure of XML data, they also have limitations where they do not cater for different types of data present in XML documents and they also cannot be easily adapted to deal with data from other data exchange formats [7], [9], [35], [81], [71], [46]. The verbosity of XML data led to the development of other compact data exchange formats, namely JavaScript Object Notation (JSON) and Binary JSON (BSON) for example [56], [38], [39], [80].

Whilst dealing with the verbose XML document structure is important to reduce the verbosity of XML data [7], [9], [35], [81], [71], [46], also finding ways to handle the content of an XML document can further improve the compression of such data.

Domain-specific data can be identified by similar data items that are contained within specific XML elements. For example in Toxicology datasets used by practitioners in the Chemoinformatics field, XML representations of this data

would include XML tags that generally relate to the names of chemicals used, description, molecular weights, and chemical linear notations such as Simplified Molecular Input Line Entry System (SMILES) for example, amongst other domain-specific data.

Delving further into domain-specific SMILES data within these datasets, this popular chemical linear notation was developed to improve storage and processing costs by representing two-dimensional molecular structures in a concise and compact way [82], [23]. However, chemical databases that contain such data are continuously expanding [42] and therefore require efficient storage and processing of such molecular structure representations [82], [23], [67].

As with XML and JSON data, general-purpose compressors could also be used over domain-specific data to allow for a reduction in compressed output size and processing costs [82], [23]. Data can also be transformed prior to data compression and used alongside general-purpose compression techniques to provide efficient data compression [69], [16], [68].

Various existing transformation techniques have been used in previous studies and show that data transformation prior to compression can improve data compressed output size and processing costs [68]. Whilst these techniques are not specific to XML or JSON data exchange formats or domain-specific data, such as SMILES, they provide the flexibility to be tailored and used with any type of data, including domain-specific data.

1.3 Aim and Objectives of the Study

The aim of this study is to investigate ways in which data transformation can be used to further improve the compression of different types of XML data, equivalent JSON data and domain-specific data, such as SMILES. The objectives of this study have been highlighted below:

- To research and select general-purpose data transforms that can be applied to XML, JSON and SMILES data, as well as data-specific transforms that can be applied to SMILES data.
- To research and select general-purpose data transform variations that can be applied to XML, JSON and SMILES data.
- To research and select general-purpose data compressors that can be applied to all data.
- In addition to the datasets commonly used in XML compression research, to research and select other datasets that would also benefit from this study.
- To transform XML, JSON and SMILES data with the selected data transforms and transform variations, and to conduct data compression experiments over the transformed data; to inform both the XML and JSON main study and SMILES case study.
- To discuss the findings of the studies, draw conclusions and discuss their implications.

1.4 Research Questions

XML and JSON Main Study Research Questions

This study aims to answer the following research questions:

- To what extent can data transformations improve the compression of different types of XML data and equivalent JSON data?
  - What general-purpose data transforms can be applied to XML and also be made adaptable to other data exchange formats, such as JSON?
  - What data transform variations can be used alongside these data transforms?

- What general-purpose compression techniques can be used over the transformed data to improve compression?
- What other datasets could be useful for this study?
- How do the final results compare for both XML and JSON data formats, in terms of providing better compressed output size, processing times and a balance of both compressed output size and processing?
- How can other existing data compression or data transform techniques can be improved?
- How far can these transforms be generalised or transferred to other data formats?

SMILES Case Study Research Questions

This case study aims to answer the following research questions:

- To what extent can data transforms and data transform variations improve the compression of SMILES data?

    - What general-purpose data transforms used in the main study, and data-specific data transforms can be applied to SMILES data?
    - What data transform variations used in the main study, and other data transform variations can be used alongside these data transforms?
    - What general-purpose compression techniques can be used over the transformed data to improve compression?
    - How do the final results compare in terms of providing better compressed output size, processing times and a balance of both compressed output size and processing?
    - How can other existing data compression or data transform techniques can be improved?

– How far can these transforms be generalised or transferred to other domain-specific data?

1.5 Significance of the Study

On a practical level, the primary contribution of this thesis is to provide developers and researchers that use XML, JSON and SMILES data with key knowledge of the best data transform and transform variation techniques to use with certain types of data, and the best compression techniques that provide the best compressed output size and processing times when used over the transformed data. The results are analysed from an industry perspective to allow developers to make decisions based on their compressed output size and processing time requirements.

On a theoretical level, another intended contribution is to suggest ways to potentially integrate the fundamental benefits of the data transforms and transform variations used in this thesis into existing XML-specific compression techniques in order to further improve them, with the purpose of making a contribution to industry and also a contribution to research.

1.6 Overview of the Study

The remainder of this thesis is divided into six chapters. Chapter 2 introduces the basics of XML data, discusses existing general-purpose data compression techniques, reviews existing XML-specific compression techniques, introduces the JSON data exchange format and compares it to XML. The chapter then focuses on domain-specific data and reviews the literature on molecular structure representations, and then describes the SMILES chemical linear notation. A literature review of existing data transformation techniques is also conducted. For both the XML and JSON main study and the SMILES case study, chapter 3

highlights the issues raised in the literature review and the general approach to the study. It then discusses the selection criteria and rationale for the datasets, data transforms, data transform variations and compression algorithms used. The chapter then explains how both the main study and case study were conducted by highlighting the data transform properties, describing the preliminary phases necessary prior to the transformation of data, highlighting the data transform phases, illustrating the grammar applied to the datasets during data transformation and briefly describing the collision handling mechanism used. It then moves onto the system architecture and implementation, and describes the full data transformation and compression process. Finally, the chapter provides an experimental assessment, including the compression metrics and benchmarks used, and discusses how the results were analysed. Chapter 4 describes the software implemented to enable users to run the data transforms and transform variations developed in this thesis. Chapter 5 presents and discusses the results of both the main study and case study, the chapter includes details on data collection, the experimental testing environment used, compression metrics, the experimental framework, gaps in the results, the results and an analysis of the results. The chapter ends with a discussion of the results. Finally, chapter 6 concludes this thesis, discussing the study implications, improvements to both existing work and this study, and future work.

Chapter 2

Background

2.1 Introduction

This chapter provides a literature review on the key areas related to the development of this thesis. Firstly, the chapter introduces the XML data exchange format, it then moves onto review and discusses the general-purpose and XML-specific compression techniques that have been used to compress XML data. The chapter then progresses to the development of the JSON data exchange format and compares it to XML data. The literature review is then focused on domain-specific data, particularly on molecular structure representations and the SMILES chemical linear notation. Research is then conducted into existing data transformation techniques that have been developed to improve the compression of data. Finally, a summary concludes this chapter.

2.2 XML

XML is a widely used data exchange format across the web. XML documents adopt the use of tags in their document structure that feature a self-describing property. Hyper Text Markup Language (HTML) also utilises the concept of tags in their document structure, however, whereas HTML documents employ tags to describe the presentation, the use of tags in an XML document is

concerned with describing the semantics of the data in order to facilitate the key functions of readability and comprehensibility of the XML documents over the web [32], [13], [46]. Figure 1 provides a simple example of an XML document to illustrate the document structure.

2.2.1 Data Model

XML documents form an ordered tree-like structure that is comprised of a series of nodes, such as element, attribute and value nodes [32], [51]. Figure 1 shows an example of an XML document. The first line of the example displays an optional prologue which essentially specifies the XML version number and encoding used, which in the example is version 1.0 and Unicode Transformation Format – 8-Bit (UTF-8) encoding respectively. The following highlights the data tree nodes present in XML documents:

- Element – Meaningful element tags are created by the user, element nodes, as shown in Figure 1 include <_id>, <name>, <awards>, to name a few. The root element is <achievements>.

- Attribute – Attributes provide further information for an element and are included within the element node itself. Whilst an example of this is not present in the example in Figure 1, the <name> element could be rewritten to include first and last names as attributes to portray the same information, for example, <name first=”John” last=”Backus”> </name>, however, this format for the name element is not recommended.

- Value – The value node denotes data values within XML documents, an example taken from Figure 1 is the “W.W. McDowell Award” value from the award element node and another example is the “1967” value taken from the year element node [32].

```
<?xml version="1.0" encoding="UTF-8" ?>
<achievements>
        <_id>1</_id>
        <name>
                <first>John</first>
                <last>Backus</last>
        </name>
        <contribs>Fortran</contribs>
        <contribs>ALGOL</contribs>
        <contribs>Backus-Naur Form</contribs>
        <contribs>FP</contribs>
        <awards>
                <award>W.W. McDowell Award</award>
                <year>1967</year>
                <by>IEEE Computer Society</by>
        </awards>
        <awards>
                <award>Draper Prize</award>
                <year>1993</year>
                <by>National Academy of Engineering</by>
        </awards>
</achievements>
```

Figure 1. Example of an XML Document adapted from [39].

2.3 General-Purpose Data Compression

Storage, query performance and efficiency issues are evident in XML documents due to the verbose document structure which has resulted in a number of compression techniques being developed to attempt to address this problem. As well as using existing general-purpose compressors over XML data, which compress the document in the same manner as any other text document, XML-conscious compressors have also been developed. General-purpose compressors do not take advantage of the XML document structure, the structure and content are treated the same, whereas in contrast XML-specific compressors utilise structure-related concepts when dealing with XML documents and many also use general-purpose compressors in their back-end processors in order to achieve optimum compression rates [66], [46].

As mentioned previously, since XML documents can be seen as text files, general-purpose data compression techniques alone can easily be used to compress these documents. A variety of compressors can specifically handle XML documents, such as XML conscious compressors that maintain full awareness of an XML document structure [46]. Structure aware XML conscious compressors include schema dependent and schema independent compressors. Although optimum compression ratios can be accomplished by utilising schema dependent compressors in comparison to schema independent compressors, the use of schemas cannot be fully relied upon as they are not present for all XML documents. Another issue to consider is how useful a schema would be for an XML document, for example, a schema may not necessarily be required for a simple XML document and it may be more useful to use a schema for a more complex XML document [66].

2.3.1 Lossy vs Lossless

General-purpose data compression is placed into two categories, namely lossy and lossless [20]. Lossy compression techniques, generally used to compress multimedia data such as audio, video and images, are unable to reproduce the document accurately on decompression. However, much of this information loss is undetectable to the end user. This is in contrast to lossless compression techniques which are able to reproduce the document exactly on decompression, thereby preserving the integrity of data [12], [68], [67]. Lossless compression can be further categorised as adaptive or non-adaptive. Non-adaptive techniques adopt two phases, firstly the computing of statistical information of the document, such as the frequency of elements in the XML document, and secondly the actual compression phase. Adaptive lossless compression techniques in contrast are a one-pass technique, whereby statistics are collected dynamically during compression, with no requirement for statistical knowledge to be collected sooner [20].

2.3.2 Statistical and Dictionary-Based Approaches

Both statistical and dictionary-based lossless compression techniques exist. Huffman is a well-known lossless compression technique through which statistical information is computed and utilised in this technique. The process starts with the frequency of the most common characters collected, the characters are then ordered according to their frequency, from highest to lowest. The sum of the frequencies of the two characters with the lowest frequencies are then computed; these two elements have been combined into one element. This process starts again with the recently grouped element and the other element of lowest frequency. A binary tree structure is assembled. The technique involves the substitution of frequent characters with shorter characters. Statistical based techniques, like Huffman coding, do not handle the connection between words and phrases [3], [16], [68], [67].

Another popular statistical compression technique is Prediction by Partial Matching (PPM). A PPM model consists of context models. These adaptively built models hold statistical information regarding the computed frequency of symbols preceding a new symbol. A statistical encoder is used to predict the probability of a new symbol. Symbols with higher probabilities are encoded with fewer bits compared to those with lower probabilities [68].

Another statistical compression algorithm is Context Tree Weighting (CTW). The data is processed as a block of symbols, and as with PPM the probability estimation is based on the statistical knowledge of the previous symbols. A binary context tree is built and updated for each symbol. The context of the symbol to be encoded determines the route node path to be followed. The values for each of the two non-leaf child nodes are updated for each symbol as well as the values of the root node. The context tree is weighted at the root node to determine the probability estimate of the processed word and then arithmetic encoding is applied [22].

Dictionary based approaches, such as Lempel-Ziv, also known as LZ77, take full advantage of repeated words and phrases within the text and encode these by substituting words or phrases with their corresponding references in the dictionary [3], [16], [62], [68] [67]. Dictionary-based approaches can also be further categorised by static, semi-static and adaptive approaches. In static-based approaches, dictionaries can be prepared in advance of compression, assuming knowledge of information contained in the document is already known. This is technically a one-pass approach as the dictionaries prepared in advance enable the compressor to use the fixed dictionary immediately. However, the static dictionaries could negatively impact compressed output size costs due to their fixed nature. Whereas, two phases are involved in a semi-static dictionary-based approach, whereby the first phase is to collect the information from the data source in order to create the dictionary, and the second phase is to proceed with the actual compression of the document. Adaptive dictionary-based approaches dynamically update the dictionary throughout compression. Although both semi-static and adaptive approaches would improve on dictionary compressed output

size costs, they may impact compression times with the data collection and dictionary updates being processed during the compression phases [16], [68], [67].

2.4 XML-Specific Data Compression

This section discusses a selection of existing XML compression techniques developed specifically for XML data. This is not intended to be a full survey, since only one of the XML compression techniques, namely XMill, discussed in this section was used in the experiments for this study for reasons that will be explained in the next chapter. A comprehensive survey of XML-specific compressors can be found in [66]. It was considered necessary to discuss some of the issues raised with XML-specific techniques in this section, in order to help understand why it was necessary to continue with research in the area of XML data compression. The inclusion of this survey also forms as a basis to provide theoretical based suggestions in the Conclusions and Future Work Chapter 6. This section was intended to provide insights into: how existing XML-specific techniques could be further improved by adopting some of the data transformation techniques discussed later in this thesis; how some of the issues raised with existing XML-specific compressors can be dealt with using these transforms; why the transforms could potentially be better from a theoretical point of view than existing XML techniques, and how some of the concepts from these techniques were adopted in the transforms and transform variations.

A host of XML-specific compression techniques have been developed and many of them use general-purpose compressors in their back-end processors. Existing XML-specific techniques can be further categorised in terms of whether or not they utilise a schema, and whether or not they have the ability to query XML documents.

2.4.1 Schema-Independent and Non-Queriable Compressors

XMill is a well-known XML compression technique that focuses on achieving good compressed output size costs. It uses a two phase approach to gather statistical information in the first phase and then applies compression in the second phase. The process involves separating the XML document structure from the content; the information contained in the structure is used to group similar items together into containers. These containers are then compressed separately. Depending on the data present in the containers, the appropriate semantic compressors are applied to compress the contents of the containers and these containers are then compressed again using GZip, thus essentially compressing the data twice. The arrangement of similar data items into the same container could provide better query processing as well as better compressed output size, if querying was permitted, as these containers would contain similar types of data and would allow for quicker access to the relevant data. Since order-preserving properties are also implied by the grouping method, this could also assist in the maintenance of data integrity on decompression [5], [9], [14], [46], [7], [45].

Other approaches are similar to XMill, in the sense that the XML structure of the document is separated from the content of an XML document. For example, XAdap groups data together based on their semantics and applies adaptive Huffman encoding. Whereby, Huffman assigns shorter codes to more frequent characters and longer codes to less frequent characters, the Adaptive Huffman algorithm includes information relating to the position of the node in the XML document and its weight [20]. Xcomp is another example of a technique that separates the structure of an XML document from its content, and groups and compresses similar items together according to their semantics. A record is kept on the original markup and data positions during the detachment of the structure part of the XML document and the contents are then assembled in order to compress similar data items together [20], [44].

2.4.2 Schema-Aware and Non-Queriable Compressors

Schemas are designed to ensure that XML documents adhere to the structural and content guidelines that have been specified in the schemas. Common examples of such schemas include Document Type Definition (DTD), XML Schema and Regular Language for XML Next Generation (Relax NG) [47]. These vary in terms of the level of information they can provide when associated with an XML document. Purely on a structural level, DTDs can support information relating to the XML elements and attributes in an XML document; however, although they have a provision whereby whitespaces can be removed, insufficient knowledge is provided on the text content of an XML document. The statistical information provided on the XML document structure is useful for data integrity purposes on decompression. Other schemas, such as XML Schema and Relax NG provide further details on the content of an XML document compared to a DTD in order to improve XML compression [19].

Millau facilitates compression by making use of information obtained from a DTD. Document structure is preserved and the transmission size is reduced. The streams generated during transmission transfer both the XML document structure and content separately. However, network and browser limitations could potentially impact processing costs [31], [88].

Schemas impose extra storage costs and techniques such as MPEG's Binary Format for Metadata (BiM) address this issue. This technique uses an XML Schema to provide information to assist in the development of an efficient binary XML document. As high storage costs are caused by the verbosity of an XML document structure, schemas provide a way to eradicate this structural redundancy [71], [70], [76]. The authors in [28] proposed a binarised text-based schema in order to cater for the use of multiple schemas, reduce storage costs and allow for schema transmission via a network. The additional overhead incurred by the pre-processing times of binary schemas could be inefficient. However, since there is sufficient information present in the schema, post-processing is not required. It is important, however, for integrity to be maintained on decompression should the

need occur to convert a binary schema back to its original form. However, in this scheme, access is only required to some areas of the schema, so only these areas are submitted to the decoder, thus reducing memory requirements [28].

BXSC is another compression technique developed for XML data streams that also uses the information contained in an XML Schema to aid its compression. The XML document structure is compressed using dynamic Huffman encoding, and other compression algorithms are used for the content of the document, depending on the types of data to be compressed. For example, incremental and dictionary based compression techniques are used for numerical and textual data, respectively [89].

2.4.3 Schema-Independent and Queriable Compressors

Several XML compressors have been developed to handle XML queries. With non-queriable XML compressors data needs to be fully decompressed to allow for any querying to be conducted on an XML document. Whereas queriable compressors allow for querying on compressed documents with either no or partial decompression. Homomorphic compressors such as XGrind discussed later (refer to Section 2.4.4) are designed to maintain the original structure of an XML document, maintaining data integrity and enabling access to the document in the same way as the original XML document. However, the original XML document structure is different from the compressed document structure in non-homomorphic compressors which may possess issues relating to data integrity [66]. Although queriable techniques provide better query processing times, they sacrifice compressed output size costs in order to achieve this. Querying over compressed data can be conducted using pattern matching by searching for compressed patterns, or by partial decompression of the compressed document [72].

XPress is an example of a homomorphic queriable compressor using simple path expressions. It enables queries to be processed over full or partially compressed documents. However, the queries are based on top-down query evaluations that impose costs on query processing times and are inefficient. Full or

partial decompression is also required for some sets of XML queries. As with XMill and other similar techniques, the compression technique is based on two phases, to gather data statistics and apply compression. These two phase scans could impact processing time efficiency [7], [9], [50].

XQZip is another technique that deals with the verbosity of structural information in XML documents, enables compression and allows for querying using XPath [9], [11]. As with XMill and other techniques, it separates structural information from the content, arranges data into containers and further segregates these into blocks which are required for partial decompression for querying purposes. GZip compression is used, and faster query processing is achieved by keeping recently decompressed blocks in cache memory. Utilising cache memory for data stored in blocks is better than main memory storage. However, issues may arise surrounding decompressed blocks that may contain data that is deemed unrelated to the query being stored in cache [9], [72].

XQueC enables query processing over compressed XML data using XQuery. In this technique, XML documents are divided into three parts, which include a structural tree made up of the structure of an XML document, data containers and a summary of the XML document structure, which consists of all distinct paths of the XML document. With path expressions already hard-coded in the containers, efficient querying is possible by parsing the structural summary instead of the XML document structure itself, reducing memory and additional processing times. However, data containers are accessed in main memory leading to potential issues with storage and processing costs. Using cache memory may improve these issues [7], [9].

2.4.4 Schema-Aware and Queriable Compressors

XGrind is a queriable compressor, using simple path expressions that also uses a DTD. However, it does not remove XML document tags that can be derived from the DTD. It uses dictionary-based encoding for XML tags and Huffman compression for the data. With no separation of structure from its content, it possesses order-preserving properties and uses homomorphic encoding, thus keeping the XML document structure and content intact after compression. Relating to query processing, exact match queries over the compressed data require no decompression of the document. However, approximate or range queries may require some partial decompression [7], [50].

Other queriable schema-aware compressors include XCQ, for example, that retain structural information from an XML document in a DTD or other schema. However, a schema is not always attached to an XML document, and if they are available they need to ensure they conform to the DTD to be used efficiently. This technique separates the structure and content from an XML document, and then deducts the knowledge contained in the schema from the knowledge contained in the structure [11], [71], [70], [8], [52].

2.5 JSON

JSON is another data exchange format that was designed as an alternative compact data interchange format to XML. As with XML, it is human readable and can easily be utilised by computers for parsing and generating purposes [56], [38], [39].

2.5.1 Data Model

Data in JSON format is held in name/value pairs with an ordered list of values. These are achieved with common data structures that are available to most programming languages. For example, the name/value pairs can be implemented in an object or hash table, and the ordered list of values can be implemented in an array or vector, amongst others. As can be seen in the example in Figure 2, name/value pairs are displayed within an object which is denoted by a set of left and right curly brackets {}. A colon is placed at the end of every name, and name/value pairs are separated by a comma. The square brackets, as with other programming languages, denote that they are holding arrays [38].

```
{
  "achievements": {
    "_id": "1",
    "name": { "first": "John", "last": "Backus" },
    "contribs": [ "Fortran", "ALGOL", "Backus-Naur Form", "FP" ],
    "awards": [
          {
           "award": "W.W. McDowell Award",
           "year": "1967",
           "by": "IEEE Computer Society"
          },
          { "award": "Draper Prize",
            "year": "1993",
            "by": "National Academy of Engineering"
          }
    ]
  }
}
```

Figure 2. Example of a JSON Document adapted from [39].

The XML version of this JSON document example was shown in Figure 1. As can be seen from examples in Figures 1 and 2, whereas XML uses both opening and closing tags, JSON does not deploy end tags and is therefore less verbose in terms of its document structure in comparison to XML. Other differences include the array usage in JSON syntax where, for example in Figure 2, the ‘contribs’ object holds an array of four contribution records and the ‘awards’ object holds an array of two awards. XML, on the other hand, as can be seen in Figure 1, defines these records in a much more verbose way by surrounding each record with opening and closing tags. Hence, JSON is much shorter and simpler to read and write compared to XML.

2.5.2 XML vs JSON

Both XML and JSON can be further compared with each other in terms of the properties highlighted below [38], [56], [40], [83]:

- Simplicity – XML has a far more verbose and sometimes complex syntax, particularly with its deeply nested structure. The syntax of JSON, in comparison, is much more straightforward. Both XML and JSON can be parsed and used by other programming languages. However, JSON has simpler direct mapping capabilities compared to XML that can easily be mapped onto data structures from other programming languages.

- Extensibility – XML was developed to be an extensible document markup language. However, JSON was not designed to be a document markup language and so the definition of new tags or attributes were not required in order to depict the data.

- Interoperability – Both XML and JSON have interoperable capabilities. This means that they can be used on a number of different platforms and their document structures can be interpreted with a variety of different parsers.

- Openness – XML has an open standard that is freely available. However, JSON is perhaps more open than XML since it is not restricted by standardization conflicts.

- Human Readability – As mentioned previously, JSON is much simpler to read and write due to its much simpler grammar in comparison to XML. This applies to both humans and computers.

- Data Exchange – As mentioned previously, both XML and JSON can be used as data exchange formats to exchange data between applications.

- Data Structure – Whilst both XML and JSON data formats provide a structure to data, as mentioned previously the structure that XML provides is far more verbose than that of JSON. XML also separates data from the structure and XML data structures include elements, attributes, data values, entities and schemas, such as a DTD, and others; data mapping could become complex particularly with deeply nested elements. However, mapping is much simpler with JSON since its data structures consist of arrays and records.

- Data Processing – XML and JSON can both be processed easily.

- Programming – Data programming is simple when handling XML data as programmers are able to reuse a wide range of existing code. JSON notation is already built into JavaScript and Python programming languages, so no extra software is required. Other programming languages only require a small amount of JSON code.

- Data Views – As XML is a document markup language, it has display capabilities, whereas, JSON does not.

- Self-Describing – The tags present in the document structure of both XML and JSON data describe the nature of the data within them (refer to Section 1.1 for a further explanation of the self-describing property within XML data).

- Internationalisation – Both XML and JSON can use Unicode.

- Data Migration – Conversion between both data formats is straightforward.

- Object-Oriented – JSON can be mapped onto object-oriented systems much more easily since it is data-oriented, whereas, XML is document-oriented.

- Widely Used – XML is more widely used in industry, however, JSON is also being used as a more compact alternative, and simple conversions from XML to JSON also make it easier to utilise JSON.

2.5.3 Other Data Formats

Other data serialisation formats exist that are comparable to both XML and JSON, for example, YAML Ain't Markup Language (YAML) and a binary-encoded format of JSON called BSON, to name a few. BSON was designed to be more compact than JSON, and MongoDB, a document-oriented database system, implemented this lightweight and fast data exchange format into its database system [87], [33], [39], [80].

2.6 Molecular Structure Representations

Over the years, the Chemoinformatics research domain has developed and implemented a number of different molecular structure representations of chemical structures. They are mainly used for the storage, representation, communication and identity checking of the chemical structures.

2.6.1 Chemical Linear Representations

In comparison to the two-dimensional or three-dimensional counterparts of the same chemical structures, the linear representations are more compact, much simpler to read and write and they can be easily inserted into software. They can also be canonical, whereby they can provide a unique representation of a molecule to allow for identity checking [57]. SMILES, as will further be discussed in the next section, is the most common of the linear representations. Its main benefit is that it is much simpler to read in comparison to other line notations, such as, International Chemical Identifier (InChI), which is designed to be machine-readable. A range of alternative linear representations include Wiswesser Line Notation (WLN), Representation of Organic Structures Description Arranged Linearly (ROSDAL), SYBYL Line Notation (SLN), Modular Chemical Descriptor Language (MCDL) and InChIKey [41], [57], [26], to name a few. Non-linear types of representations include MOL and SDF formats, designed for single and multiple molecules, respectively, which contain information relating to the coordinates and connections between atoms [41]. The linear notation, SMILES, uses approximately between 50% and 70% less storage compared to the corresponding connection tables [23], [41], [57], [26], [67].

Real Toxicology data was being used in other contexts by researchers in the department working on other research projects. It was therefore suggested by M. Ridley that the semi-structured nature of this data would prove to be useful if it were to be used as a case study in this research [supervision meeting, 12 July

2012]. With the continuously expanding chemical databases and the popularity of chemical linear notations, such as SMILES, efficient storage and processing of such molecular structure representations are necessary. Whilst the other chemical notations mentioned could also have been used as a case study in this research, SMILES was selected due to its popularity amongst its users and also due to its interesting grammar. Further details of the rationale for selecting SMILES as a case study can be found in Section 3.3.1 [42], [82], [23], [67].

2.6.2 SMILES

Simplified Molecular Input Line Entry System (SMILES) is a linear chemical notation widely used by practitioners in the Chemoinformatics field to represent molecular structures. It was originally developed by Weininger in 1988 [82] and extended by Daylight Chemical Information Systems [23]. SMILES contains the same information from its equivalent connection table, but with a much simpler vocabulary in terms of atom and bond symbols and grammar rules (as described in Table 1) [82], [23], [67].

Table 1 provides a guide to the basic grammatical rules for SMILES notations along with some examples. ASCII characters are used to represent SMILES notations in linear format with no spaces permitted. Whilst molecular structures can have several different and valid SMILES representations, this means that SMILES strings are not unique since there can be more than one SMILES representation of a molecular structure. However, unique generation of SMILES is possible using canonicalisation [82], [23], [67].

As can be seen from Table 1, atoms are represented by atomic symbols. Atoms displayed as being enclosed within square brackets denote that non-organic atoms are present; other important information contained in the brackets include the number of Hydrogen atoms and atomic charges. However, the brackets are not required when representing organic atoms. Hydrogen atoms are understood to be present for these atoms even with the square brackets omitted. Atoms represented in uppercase characters are known as aliphatic atoms and atoms that are depicted

in lowercase characters are commonly known as aromatic atoms [82], [23], [67].

Adjoining atoms are assumed to be connected to each other by single bonds or aromatic bonds; these bonds can be excluded from the notation. Double or triple bonds, however, must be displayed. Nested or stacked branches are parenthesised. Representation of a cyclic structure or ring closure involves breaking a single bond inside a cyclic ring, and then the ring opening and closing are determined by the numbers included after the ring opening and ring closing symbols. The separation of disconnected structures are represented with a period [82], [23], [67].

Table 1. Generic SMILES Representation Rules and Examples [82], [23], [67].

Generic SMILES Rules	Example Representations
Non-Organic Atoms	[S], [H+]
Aliphatic Organic Atoms	B, C, N, O, P, S, F, Cl, Br, I
Aromatic Organic Atoms	b, c, n, o, p, s
Single Bonds	C-C, CC
Double Bonds	C=C
Triple Bonds	C#N
Aromatic Bonds	c:c, cc
Nested or Stacked Branches	C=CC(CCC)C(C(C)C)CCC
Ring Closures	C1CCCCC1
Disconnections	[Na+].[0-]c1ccccc1

2.7 Data Transforms

Data transformation techniques have similar goals to the XML-specific compression techniques discussed previously. They are able to improve compression by applying them to the data prior to compression. However, the transformation phase prior to compression in XML-specific techniques are generally focused on the structure of XML documents. In order to achieve the required compression, data transformation techniques consist of additional pre-

processing and post-processing steps, which can lead to an increase in both compressed output size and processing times. However, it is possible to achieve the desired compression results with the use of both data modelling and general-purpose compression techniques [69], [16], [68] [67]. A number of existing data transformation techniques have been described in [60], [64], [68]. The following sections discuss existing data transformation techniques.

2.7.1 Burrow-Wheeler Transform

The Burrows-Wheeler Transform (BWT) uses a Block Sorting algorithm, and a combination of Move-To-Front (MTF) and Huffman or arithmetic encoding. The process involves the dividing of the input text into blocks of identical lengths, using the Block Sorting algorithm to rearrange these blocks, creating clusters of similar symbols and then applying encoding techniques such as MTF alongside Huffman or arithmetic encoding. As the Block Sorting algorithm performs cyclic shifts among the original string and sorting operations among the resulting strings, this is considered to be the most time consuming phase in the process as the output is ordered lexicographically [60].

2.7.2 Star Transform Schemes and LIPT

In the Star Transform encoding scheme [64], a large fixed dictionary, created from the input text file, consists of the most frequently used words anticipated from the text file and their star encoded equivalents. The star encoding scheme substitutes letters using the star '*' character, with the most frequently used words containing the greater number of star '*' characters, and a smaller number of star '*' characters assigned to the less frequent words [64]. This encoding scheme also utilises the capital conversion method, that will be discussed later, whereby tokens are used to signify whether the word starts with a capital letter and is then followed by a sequence of lowercase letters, or whether

the word consists purely of capital letters. The tokens used in this capital conversion method include a collision handling mechanism. Compression is improved since the star '*' character is predominantly the main character in this transformation [68].

Whilst the Star Transform encoding scheme focused on the frequency of words, Length Index Preserving Transform (LIPT) made use of both the frequency of words and the word lengths. LIPT was one of the transforms developed to improve on the Star Transform encoding scheme. In this transformation technique, the encoding consists of a single star '*' character preceding each codeword, which also signifies the start of the codeword. This is followed by the word length and the index. In comparison to the Star Transform encoding scheme, LIPT uses shorter codewords for frequently used words and the codewords are generally created smaller in length compared to the original words. Lexicographical word sorting in the dictionary, and binary search operations performed in the encoding and decoding stages of LIPT, improved processing speeds, as did the introduction of random access to words in the decoding phase, compared to the Star Transform encoding scheme where word searches in the encoding and decoding stages were inefficient [68], [60].

The Star New Transform (StarNT) is based on the LIPT transform. However, whereas LIPT includes the original word length into the codewords, StarNT omits the word length, and where the star '*' character preceding the word length in LIPT signifies the start of the codeword, the star '*' character in StarNT simply indicates that the word could not be found in the dictionary. Hence, since the star '*' character is being used in a different context in the StarNT transform compared to the LIPT transform, less of these characters are used compared to LIPT, thus increasing compression effectiveness. Essentially, this means that StarNT only consists of a dictionary index [68].

2.7.3 Capital Conversion

The capital conversion technique is a simple data transformation technique designed to substitute capital letters at the start of the word with their equivalent lowercase letters, using a flag to signify the change. Using another flag can be beneficial to convert words to lowercase that consist purely of all uppercase letters. However, these are limited to substitutions of words that consist of the first letter capitalised, or cases where the whole words are capitalised; situations in which words contain capital letters in positions other than those mentioned have not been considered. For example, some of the compression algorithms that will be discussed in the next chapter, namely, '7Zip', 'BZip2', 'GZip' and 'PPMd' consist of a capital letter that occurs after the first character for 7Zip, two consecutive capital letters at the beginning of the words for BZip2 and GZip and three consecutive capital letters at the beginning of the word for PPMd. Whilst it does have its limitations, the former technique is justified with its prediction capabilities, whereby the word starting with a capital letter is assumed to be at the beginning of the sentence, thereby allowing for flag prediction with the assumption that this is the first non-whitespace character following a terminal punctuation mark, such as a period, question mark or an exclamation mark. Whilst the first capitalised letter of a sentence is easy to predict based on these assumptions, the second lowercase letter is difficult to predict, which means the general-purpose data compressor names mentioned above would be harder to predict completely as they violate some of these assumptions with some capital letters occurring elsewhere in the sequence of characters. However, the predictions described do increase contextual dependencies and word similarities [68].

2.7.4 Space Stuffing and End-of-Line Encoding

Space stuffing is a technique that uses the knowledge that lines within a text document end with Linefeed (LF), Carriage Return (CR) or CRLF End-of-Line (EOL) symbols instead of spaces. These symbols are normally followed by a word starting on the next line. However, as words normally follow on from spaces in a text document, word forecasting is not always possible, particularly if a word follows on from both space and EOL symbols. The notion of space stuffing is to include spaces at the start of every line in the document, following on from EOL symbols as opposed to preceding them, thus, improving the prediction of words [68].

The reasons for developing the End-of-Line (EOL) encoding technique are similar to those mentioned in the space stuffing transformation technique, mainly due to the inability to predict words. As with space stuffing, EOL encoding also substitutes EOL symbols with spaces. However, this technique also encodes details that allow for the reverse operation separately [68].

2.7.5 Punctuation Marks Modelling

Punctuations marks modelling was developed from the notion that punctuation marks follow on directly from words, and do not follow on from spaces in text documents. Thus, indicating that character prediction is more difficult with this notion, and that this can simply be solved by the addition of a space symbol preceding a punctuation mark in order to allow for better prediction of characters. Also, situations whereby spaces are already present before a punctuation mark prior to transformation, have their spaces removed. This technique is very similar to the space stuffing idea [68].

2.7.6 Alphabet Reordering

An alphabet reordering technique was proposed by Chapin and Tate [18] as an improvement to the BWT transform mentioned earlier. Several alphabet reordered combinations were examined, and the combinations that started with vowels followed by similar consonants and punctuation marks, such as ‘aeioubcdfghjklmnpqrstvwxyz’ or another reordering example based on uppercase letters, ‘AEIOUBCDGFHRLSMNPQJKTWVXYZ’, a similar reordering was developed for lowercase letters; provided a further improvement to compression [64], [68]. However, these reorderings are best on textual files that contain a lot of alphabetical characters, compared to alphanumeric and numeric characters, although the reordering could be extended to incorporate these. Abel and Teahan [2] had improved on this further by including numeric characters and punctuation symbols in their reordering technique. Their alphabet order for capital letters, which was also similar to lowercase letters, was ‘SNLMGQZBPCFMRHAOUIYXVDKTJE’. Notice in this case, the location of the vowels has mainly moved towards the centre and the ‘E’ character is placed at the end in this reordering [68], [2], [18].

2.7.7 Character and Word N-Gram Transforms

A character n-gram (also known as a q-gram) consists of a string of n consecutive characters. However, an n-gram can also be used to refer to a string of non-adjoining characters, such as, the first and fourth characters in a string; the term n-gram used in this thesis will refer to adjoining characters. Examples of two, three and four character n-grams based on the word ‘transform’ are as follows:

- Character Bi-gram (or Di-gram): tr, ra, an, ns, sf, fo, or, rm
- Character Tri-gram tra, ran, ans, nsf, sfo, for, orm
- Character Quad-gram tran, rans, ansf, nsfo, sfor, form

The same concept is used for word n-grams, except that they refer to contiguous words, that are identified by the fact that they are separated by space symbols, instead of characters. Examples of two, three and four word n-grams based on the sentence ‘this is a data transform’ are as follows:

- Word Bi-gram: this is, is a, a data, data transform
- Word Tri-gram this is a, is a data, a data transform
- Word Quad-gram this is a data, is a data transform

In this technique, frequently used character and word n-grams are gathered and placed into a dictionary whereby substitutions can be executed during the transform. Fixed dictionaries can be used if the n-gram information is already known, whereas semi-static dictionaries require an additional pass of the text document in order to collect n-gram frequencies. N-grams, particularly bi-grams, improve compression as they reduce the size of the document. Evidently, word n-gram transformations will provide greater file size reductions compared to character n-gram substitutions [17], [61], [68].

2.7.8 Word-Based Data Transforms

Word-based data transformation techniques simply substitute words with shorter codewords and can provide far better compression results compared to other text based pre-processing methods. As mentioned previously, using a fixed dictionary with the words from the text file already inserted into the dictionary, is faster to process compared to that of a semi-static dictionary, which requires an additional scan of the text file in order to collect the word frequencies and build the dictionary. Using a fixed dictionary also avoids the issue of having to transmit the dictionary together with the output data [68].

Semi-static dictionaries have been used in the semi-static word replacement approach, proposed by Abel and Teahan [2]. In their technique, common words are substituted by unused characters which also form the dictionary indexes. However,

they do not consider the substitution of words with existing characters. The process involved making decisions on which unused tokens were available, which unused tokens were reserved for frequent tri-grams and bi-grams, and which of the remainder unused tokens could be used for word substitution. At the same time, word frequencies were computed on word lengths that consisted of two or more words. The next step involves computing which words occur in the input text with a frequency of at least 25%, computing their weights based on the word frequencies and word lengths, and then sorting these words in descending order, with the highest weighted words being assigned the available unused tokens, as long as they have a weight of at least 16. The final stage involves the actual substitution of the words with the unused tokens. The semi-static dictionary approach enables the dictionary to be reconstructed by substituting a word that has occurred for the first time with an unused token, followed by the actual word, and then later when the word reoccurs, it is simply substituted by the token in the dictionary [68].

The Word Replacing Transform (WRT) was developed by Skibiński to transform text based on the dictionary-based and word transforms previously mentioned. This technique incorporated a number of related data transforms into its method, namely, capital conversion, sorting according to word frequency, character n-gram (or q-gram) replacement, and End-of-Line (EOL) symbol to space replacements. It also consists of data protection, such as data filtering, and utilises the space stuffing technique amongst words. The authors extended this technique, WRT, to Two-Level Word Replacing Transform (TWRT). This extension permitted data-specific dictionaries to be used in the substitution process. For example, the inclusion of a dictionary possessing words specific to a Java programming language has the potential of enhancing compression [69], [34], [68], [67], [64].

2.7.9 XML-Specific Data Transforms

A data transformation technique that was specifically developed for XML data, known as the XML Word Replacing Transform (XWRT), was developed to provide efficient compressed output size and processing of compressed XML data. As with other techniques mentioned, this was also a dictionary-based technique, with frequently used words being substituted by their corresponding references in the dictionary. As it uses a semi-dynamic dictionary approach, the dictionary can be reconstructed with new words accordingly. It also handles frequently used alphanumeric phrases. It is also possible in this technique to omit spaces before the encoding phase and then restore these spaces during the decoding phase. However, data integrity could be an issue here on decompression with spaces being removed and then re-inserted. XWRT is a lossless technique that makes use of cyclic redundancy check and hash functions in order to keep the integrity of a document in check. The authors achieved faster decompression with this technique and it was designed to be a fully reversible data transformation technique [71], [70], [72].

QXT is another fully reversible data transformation technique developed specifically for XML data. As with XWRT and other semi-dynamic dictionary approaches, it substitutes frequently used words in an XML document with its corresponding indices and encodes these indices. As with some of the XML-specific compression techniques mentioned previously, it also arranges data with similar structural contexts together and adds these to separate containers. It is also a queriable technique that permits querying over compressed data with only partial decompression necessary, without having a severe impact on compression ratios. However, as it uses a semi-dynamic dictionary approach, as mentioned in previous techniques, it involves two pass scans. The first pass involves statistical gathering of the word frequencies and dictionary creation; in order to allow for faster decompression the dictionary is transmitted alongside the compressed file. The actual data transformation is the main process in the second pass, including data parsing and encoding, and the arrangement of similar data according to their

structural contexts into their designated containers. Whilst the two pass scans are necessary in order to ascertain the frequencies of words for statistical gathering purposes, this could impact processing time and efficiency [72].

2.8 Chapter Summary

This chapter provided a literature review on data exchange formats, data compression techniques, domain-specific SMILES data and data transformation techniques. The following points can be concluded from this chapter:

- Whilst being a common data exchange format, XML documents possess a verbose document structure that negatively impacts data storage and processing times. The problematic redundancy of XML data resulted in the development of other data exchange formats designed to be compact alternatives to XML, such JSON.

- General-purpose compression techniques have been used but were not considered suitable for XML documents as they treated XML documents as a text file. This resulted in XML-specific compression techniques being developed, tailored specifically for XML data.

- Data transformation techniques have also been developed to provide efficient compression over data when used alongside general-purpose compressors.

- Efficient data storage and processing has also been identified as necessary for molecular structure representations, such as SMILES data.

The next chapter discusses how this study will be conducted.

Chapter 3

Methodology

3.1 Introduction

The previous chapter provided a literature review of key areas relevant to this research. In particular, it discussed the issues relating to XML data, general-purpose and XML-specific data compression techniques used to deal with XML data, alternative data exchange formats, such as JSON, domain-specific data, such as SMILES, and data transformation techniques that have been developed to also provide compression of data. The issues highlighted in the previous chapter formed the basis of the research design described in this chapter.

This chapter aims to draw on the conclusions from the previous chapter to discuss the methods to be used to answer the research questions stated in Section 1.4. The key components of this chapter include the following:

- Selection criteria and rationale for datasets, data transforms, data transform variations and compression algorithms used.
- Data transform properties that the selected transforms are expected to hold and the techniques used in the selected transform phases, including the preliminary phases.
- Data transform grammar applied during the transform phases and a brief description of the collision handling mechanism used.
- System architecture and implementation, including a full description of the data transform and compression process.

- Experimental assessment of the testing environment, compression metrics and benchmarks used.
- Results analysis tools and procedure.

Finally, a summary concludes this chapter.

3.2 Main Research Study

3.2.1 XML and JSON

Research Study Observations

The conclusions derived from the literature review conducted in the previous chapter highlighted the need to further investigate data compression for common data exchange formats, particularly XML and JSON. This was based on the following observations identified from the literature review:

- XML is a widely used data exchange format that has been used in industry for many years. Whilst the composition of an XML document provides users with sufficient information in order to process such documents effectively, its verbose structure leads to issues relating to data storage and processing times.

- The verbosity of XML data also resulted in the need to develop alternative and more compact data exchange formats, such as JSON; which is quickly being recognised in industry and is in direct competition with XML.

- In an attempt to resolve the storage and processing issues cause by the verbose XML structure, researchers and developers used widely available general-purpose compression techniques. These compressors did provide some relief for both compressed output size and processing costs.

However, they were designed for text files and therefore treated an XML document as a text file during compression.

- As an XML document is not considered a general text file, the use of general-purpose compressors over XML data was not considered suitable for this data format. This reasoning led to the development of a variety of XML-specific compression techniques, that made good use of the information contained in the structure of an XML document structure in order to provide adequate compression, and also utilised existing general-purpose compression algorithms in their back-end processors. However, the following issues were identified with these compressors:

  - With the exception of XMill and a few others, many of these XML-specific compressors are not widely available.
  - Whilst many had been tested on industry-wide XML datasets, the tests could be extended to include datasets from other disciplines.
  - They are not flexible towards other data formats, so they can only be applied to XML data.

- A number of data transformation techniques have been developed in the past that can be used with general-purpose compressors to efficiently compress data by applying general-purpose compression techniques over the transformed data. These transform techniques can be easily developed and tailored towards the data being compressed. However, the following issues were identified with these transforms:

  - Some of these data transforms had not been tailored or applied to XML or JSON documents; with the exception of XWRT that had been tailored to XML data and could also be used with any type of data according to the author [71], [70], [72]. However, they had not considered data transform variations that have been mentioned

below (refer to Section 3.2.4).

– Whilst some areas reviewed in the literature did reveal research conducted in data transform variations (refer to Section 2.7), a comprehensive experimental review of using unused and existing characters, alphabetic, alphanumeric and numeric characters, and uppercase and lowercase characters had not been considered.

XML and JSON Main Research Study

In order to provide efficient compression of the XML data exchange format, one needed a solution that resolved the issues raised with existing XML-specific compression and data transformation techniques, as mentioned above. The suggested solutions to these key observations, which have been highlighted below, formed the basis of the main research study carried out in this thesis in order to answer the research questions highlighted in Section 1.4:

- To use general-purpose data transforms that can easily be developed and implemented by developers, and be flexible for transferability so that they can be applied to other data formats, for example, XML and JSON data formats.

- To investigate data transform variations that can be used alongside these data transforms.

- To use widely available general-purpose compression algorithms over the transformed data.

- To carry out experiments on datasets that are considered to be industry-wide, deemed more useful in a practical sense.

- On a practical level, to analyse all observed results from an industry perspective in order to allow developers to make more informed decisions

on the best data transform, data transform variation and data compression algorithm to use on the required type of data, based on their requirements, including: better compressed output size costs; better processing times, and a balance of both compressed output size and processing costs to provide information relating to computing resource requirements for their selections.

- On a theoretical level, to discuss ways of integrating these data transforms, in some form or other, into existing XML-specific compression techniques in order to further improve them.

3.2.2 Data Collection

The criteria for the selection of datasets that would be useful in the XML main study were set as follows:

- To represent industry-wide usage.
- To be easily convertible to XML if they were not already in this format.

As well as using many of the well-known industry-wide XML datasets, commonly used in XML compression research, the XML data corpus which are comprised of the XML datasets used in this study was further extended to include data from the Toxicology discipline. Whilst data from other areas could also have been considered, Toxicology data was of interest due to its semi-structured nature and variety of different types of data held within the datasets. However, many of these datasets needed converting to XML. Whilst there are many tools to facilitate the conversion of datasets to XML, the simplicity and availability of Microsoft Excel was sufficient enough for this purpose. The same datasets were converted to JSON using Oxygen XML Editor [58] to be used for the JSON set of the experiments. Where the XML files were too large to be converted to JSON, these files were split using FileSplitter [30], converted to JSON and then manually merged together. Descriptions of the datasets used in these experiments can be

found in Table 2 below, and XML and JSON dataset sizes and characteristics can be found in Appendices A and B, respectively.

Table 2. Dataset Descriptions

Data Group	Description
Auction [86]	Auction data
Courses [36]	Collection of university courses
DBLP_SIGMOD [86]	ACM SIGMOD record articles
DSSTox [24]	Toxicology data
EXI [25]	Collection of data from the EXI interoperability framework
ExpoCast [27]	Toxicology data
NASA [86]	Astronomical data
PSD_SwissProt [86]	Protein sequence database
QSAR [59]	Toxicology data
RDF [63]	RDF data
RNA [65]	Collections from the noncoding RNA database
Shakespeare [37]	Shakespeare plays
ToxCast [77]	Toxicology data
TPC-H [86]	TPC-H relational database benchmark
Treebank [86]	Partially encrypted annotated text for linguistic structure
Wikimedia [84]	Collection of Wikimedia wikis
XBench_XMark [85]	Data-centric and text-centric datasets

3.2.3 Data Transforms

The selection of the data transform techniques to be used in this study was based on the following criteria:

- To be suitable for data consisting of both structure and content.
- To allow for flexibility and adaptability to any data format.

Tag Conversions

Whilst XML-specific compression techniques were not used in this study, with the exception of XMill, for reasons that will be discussed later (refer to Section 3.2.11), the tag data transform idea that was generated by some of these

techniques relating specifically to the XML document structure, was used in this study. This was a simple idea to just transform the verbose XML document structure into a less verbose structure, by substituting tag names with shorter codewords. It also maintained homomorphism, which was explained earlier; this essentially keeps the document as an XML document. As homomorphism maintains the XML document structure, this has many benefits, including enabling querying over the transformed XML document with XML-specific query languages, such as XPath and XQuery. Also, data integrity is maintained on decompression since the XML document structure and data remain together, unlike some of the other XML-specific compressors that separate data structure from content in their techniques, such as XMill [5], [9], [14], [46], [7], [45]. Whilst the separation of structure and data does have compression benefits, and they do also group similar data together in containers, ensuring that data integrity is maintained on decompression would be important here to ensure that the document is reproduced accurately back to its original form. The same tag conversion technique was applied to the JSON datasets for consistency and comparative purposes.

Capital Conversions

The capital conversion techniques [68] allow for word similarities to emerge, and involve the substitution of uppercase characters with their corresponding lowercase counterparts. Whilst this data transform provides more benefits over data that contain more alphabetical characters than alphanumeric or numeric characters, with the verbose structure of XML documents, where the content of an XML document may not benefit from this transform, the tag names in the XML document structure would potentially still benefit from using this data transform. Therefore, this was considered a suitable transform for this study. Also, the same capital conversion technique was applied to the JSON datasets.

Character N-Gram Substitution

Character n-gram substitution [68] was included in this study. The technique essentially substituted frequently used substrings of n consecutive characters. The n-gram selections included those from the XML and JSON document structure and content. Also, unlike capital conversion, n-gram substitutions are not limited to the type of data, for example, alphabetic, alphanumeric or numeric. They can be computed for all characters in the dataset.

Word N-Gram Substitution

Similarly, word n-gram substitution [68] was also included in this study. This technique involved the substitution of frequently used substrings of n consecutive words. As with character n-grams, word n-grams also included those from the XML and JSON document structure and content, and can be computed for all words in the datasets. The benefits of word n-grams can be visible in terms of compressed output size.

In order to assist this particular data transform, the techniques relating to space stuffing and EOL encoding [68] were used in conjunction with word n-grams. To facilitate word prediction in word n-gram collection, preliminary data transforms were applied over the XML and JSON datasets. These included converting EOL symbols to spaces and also adding spaces after tags, since both word prediction and n-gram word collection rely on words being surrounded by spaces, otherwise they are treated as one word which would result in inaccurate n-gram collections.

Punctuation marks modelling [68] was initially considered for this study but not used due to the excessive run times of the other data transforms used in this study. Since this technique uses the same principles as the space stuffing technique – it improves word prediction by including a space before a punctuation mark – it could be used in the future to further improve this study. Particularly, in terms of word n-gram collection, the inclusion of this data transform would add further insights into the results from the word n-gram data transform, which will be discussed later.

3.2.4 Data Transform Variations

In addition to the data transform criteria set in the previous section, the selection of the data transform variation techniques to be used in this study was based on the following criteria:

- To provide variations that can be applied to the chosen data transforms.
- To provide varied combinations that can be used in conjunction with other transform variations.

Existing and Unused Variation

The Star Transform encoding scheme was described in the literature review by [64] and [68]. This particular transform was not used in the main study undertaken in this thesis, due the concerns relating to the expected increase in transformed file sizes with this encoding scheme, particularly with larger files. However, the concept of reusing existing characters that was used in this study, was taken from this method. As previously mentioned, the star encoding scheme converts frequently used words with greater numbers of the star '*' character, thus, essentially filling up the data with this single reused character. Whilst the size of the transformed data expands, improvements in compression can be achieved with this technique. In this study, reused characters were based on the most frequent characters generally across XML datasets. Unused characters in these data transform variations were based on unused characters across all XML datasets. The same technique was applied to the JSON datasets.

Alphabetical, Alphanumeric and Numeric Variation

A data transform variation that included a wide set of characters, such as alphabetic, alphanumeric and numeric characters was required to provide further comparisons. The alphabet reordering techniques developed by Chapin and Tate [18] and Abel and Teahan [2] were the influence for the alphanumeric part of this data transform variation. The authors described how the rearrangement of the

vowels first, then followed by the consonants, provided an improvement in data compression [18], [2], [64] and [68].

Uppercase and Lowercase Variation

The concepts of the capital conversion technique [68] in terms of converting uppercase to lowercase letters was used in this transform variation. However, as well as using lowercase and uppercase substitutions, a combination of both were also used.

3.2.5 General-Purpose Data Compressors

The selection of the general-purpose data compression techniques to be used in this study was based on the following criteria:

- To be easily available.
- To be widely-known to industry.
- To represent varied compression techniques.

The general-purpose compression techniques, shown in Table 3 were applied to all transformed data in this study. Some of these techniques have been used as compression algorithms in back-end processors of XML-Specific compressors, such as XMill [5], [9], [14], [46], [7], [45]. Many of these general-purpose compressors have also been used in previous XML compression research in order to provide comparisons with XML-Specific techniques developed [66]. As can be seen from the selection: 7Zip uses a combination of the Lempel-Ziv, Huffman, PPM and BWT techniques described in Sections 2.3.2 and 2.7.1 respectively; BZip2 uses both BWT and Huffman encoding also described in both sections; GZip is based on both Lempel-Ziv and Huffman encoding which were discussed in Section 2.3.2; and PPMd, PPMVC and ZPAQ are all based on the PPM technique mentioned in Section 2.3.2.

Table 3. General-Purpose Compressors adapted from [67]

Compressors	Compression Algorithms/Techniques
7Zip [62], [1]	Back-End: LZMA (Default), LZMA2, PPMd, BZip2, DEFLATE
BZip2 [62], [15]	Uses BWT and Huffman
GZip [62], [75]	Based on DEFLATE (LZ77 and Huffman)
PPMd [68], [21]	Based on Prediction by Partial Matching (PPM), Adaptive Statistical Technique using Context Modelling and Prediction
PPMVC [68], [34]	Based on PPM with Variable-Length Contexts
ZPAQ [90]	Back-End: LZ77 (Default)

3.2.6 Data Transform Properties

Whilst different data transforms have been used and this study is to carry out a comparative analysis of them all, the following properties are expected to be held by the data transforms developed in the main study; these properties were also used in the case study published in [67]:

- Reduced Compressed Output Size – Compressed output size is expected to be reduced when general-purpose compression techniques have been applied to the transformed data.
- Reduced Processing Times – The time taken to compress and decompress the transformed files with general-purpose compressors are expected to be reduced.
- Collision Handling – Collision handling is necessary to enable data to be fully reversible on decompression, thus preserving data integrity. This is particularly important with the existing character data transform variations used in this study.

3.2.7 Data Transform Preliminary Phases

The following was carried out prior to applying data transforms and transform variations over both XML and JSON datasets:

- Character Encoding – After XML datasets had been obtained from their sources, and those that required converting to XML were converted (in order to ensure all datasets contained the same character encoding) they were all converted to UTF-8 without BOM character encoding using Notepad++ [55]. The recommendations for not using BOM was based on [79]. The same character encoding was applied to the JSON datasets after the conversion from XML to JSON data.

- Distinct XML Tags – In order to determine tag tokens to be replaced in the tag data transform, the XML documents were queried with XQuery to obtain the distinct tag names. The same distinct tag names were used in the tag data transform for the JSON datasets.

- Character N-Gram Collection – To determine token replacement prefixes to be used in some of the data transform variations, and also to establish the character n-grams to replace, the following steps were carried out using the n-gram collection tool, Text2Ngram [54], on all XML and JSON datasets:

  - One character n-grams were collected and sorted according to their frequencies in order to determine the most frequently used non-alphabetic and non-numeric UTF-8 characters that could be used as prefixes for the existing data transform variations in the capital conversion, character and word n-gram data transforms. It was also used as a guide to analyse which characters were not used, and this information was needed to establish unused prefixes that could be used in the data transform variations for these transforms.

- Two to ten character n-grams were collected and sorted according to their frequencies to determine the most frequent character n-grams that required token replacements in the character n-gram data transform. The top 26 most frequent character n-grams were selected as the character n-gram tokens that needed replacing. The number 26 was purely based on the number of letters in the alphabet as alphabetical characters were being used in the data transformation variations. A small case study was carried out on five XML files of varying sizes from 1 KB to 58819 KB to ascertain if the number of most frequent character n-grams used affected data compression. The study tested 10 to 50 most frequent characters using the two character n-gram data transform with the Existing_Number transform variation. The study found that compression ratios were slightly better when the number of frequent characters to substitute were higher. The decision to substitute the top 26 most frequent characters, which is mid-range, seemed appropriate for this study. Future work will involve using other numbers of frequent characters to substitute and this number could also be varied for different types of data.

- Word N-Gram Collection – To determine the word n-grams to replace, the following steps were carried out on all XML and JSON datasets:

  - As mentioned earlier, in order to better facilitate and provide more accurate word n-gram collections, EOL characters were converted to spaces using Notepad++, and spaces were also added after the XML structure tags on all datasets.
  - Text2Ngram was used to collect information relating to one to five word n-grams to determine the most frequent word n-grams that needed token replacements in the word n-gram data transform. As with character n-grams for consistency purposes the top 26 most frequent word n-grams were selected as the word n-gram tokens that

needed replacing.

3.2.8 Data Transform Phases

The following shows the techniques used in the different transform phases:

- Tag Conversions – Distinct element tag names were replaced with alphabetic, alphanumeric, lowercase and uppercase characters. The replacement characters used were enclosed within the original start and end tags.

- Capital Conversions – Uppercase characters were replaced with their lowercase equivalent characters, which were also preceded with either an existing or unused character to mark the location of the uppercase characters. This technique enabled the conversion of all uppercase characters in the document; it was not just limited to words that started at the beginning of the sentence or words that contained just capital letters as was the case in previous studies. The number of prefixes preceding letters in words that contained just capital letters could have been reduced by preceding these kinds of words with a separate prefix at the start of the word. However, this was not included in this study as it would require an extra pass over the data to determine these sets of words. Whilst an extra pass was not considered necessary for this data transform, an investigation into how these sets of words can further improve data compression can be another area for future work [68].

- Character N-Gram Transformations – Frequent tokens from two to ten characters in length were replaced with a variation of alphabetic, alphanumeric, numeric, lowercase and uppercase characters, which were also preceded by prefixes that used either existing or unused characters.

- Word N-Gram Transformations – Frequent tokens from one to five words in length were replaced with a variation of alphabetic, alphanumeric, numeric, lowercase and uppercase characters, which were also preceded by prefixes that used either existing or unused characters.

3.2.9 Data Transform Grammar

Table 4 illustrates the grammar applied to both the XML and JSON datasets during the data transform phases. Further descriptions of the data transform variations can be found in Section 5.3. The following are examples of XML and JSON data and their equivalent transform representations to illustrate the transform scenarios further:

XML

- Tags
  - Untransformed: <Tag1> This is a test </Tag1>
  - Lowercase: <a> This is a test ; <aa> This is a test </aa>
  - LowerNumbers: <a1> This is a test </a1>; <a2> This is a test </a2>
  - LowerUpper: <a> This is a test ; <aA> This is a test </aA>
  - Uppercase: <A> This is a test </A>; <AA> This is a test </AA>
  - UpperLower: <A> This is a test </A>; <Aa> This is a test </Aa>
  - UpperNumbers: <A1> This is a test </A1>; <A2> This is a test </A2>
- Caps
  - Untransformed: <Tag1> This is a test </Tag1>
  - UnusedPrefixes: <|tag1> |this is a test </|tag1>
  - ReusedPrefixes: <//tag1> //this is a test <///tag1>
- Two-Char N-Gram
  - Untransformed: <Tag1> This is a test </Tag1>
  - Existing_Lowercase: <<m<<a<<d <<l<<c <<c a <<j<<g <<<h<<a<<d
  - Existing_Lowernumber: <e3<a1<a4 <e2<a3 <a3 a <a0<a7 <<a8<a1<a4

- Existing_Caps: <<M<<A<<D <<L<<C <<C a <<J<<G <<<H<<A<<D
- Existing_Uppernumber: <E3<A1<A4 <E2<A3 <A3 a <A0<A7 <<A8<A1<A4
- Existing_Numbers: <13<1<4 <12<3 <3 a <10<7 <<8<1<4
- Existing_Unused: <£<ƒ<Œ <¢<Š <Š a <Ÿ<š <<œ<ƒ<Œ
- Unused: £ƒŒ ¢Š Š a Ÿš <œƒŒ

■ Two-Word N-Gram

- Untransformed: <Tag1> This is a test </Tag1>
- Existing_Lowercase: <Tag1> <<b <<a </Tag1>
- Existing_Lowernumber: <Tag1> <a2 <a1 </Tag1>
- Existing_Caps: <Tag1> <<B <<A </Tag1>
- Existing_Uppernumber: <Tag1> <A2 <A1 </Tag1>
- Existing_Numbers: <Tag1> <2 <1 </Tag1>
- Existing_Unused: <Tag1> <ˆ <ƒ </Tag1>
- Unused: <Tag1> ˆ ƒ </Tag1>

JSON

■ Tags

- Untransformed: { "Tag1": " This is a test " }
- Lowercase: { "a": " This is a test " }; { "aa": " This is a test " }
- LowerNumbers: { "a1": " This is a test " }; { "a2": " This is a test " }
- LowerUpper: { "a": " This is a test " }; { "aA": " This is a test " }
- Uppercase: { "A": " This is a test " }; { "AA": " This is a test " }
- UpperLower: { "A": " This is a test " }; { "Aa": " This is a test " }
- UpperNumbers: { "A1": " This is a test " }; { "A2": " This is a test " }

■ Caps

- Untransformed: { "Tag1": " This is a test " }
- UnusedPrefixes: { "|tag1": " |this is a test " }
- ReusedPrefixes: { ":"tag1": " :"this is a test " }

■ Two-Char N-Gram

- Untransformed: { "Tag1": " This is a test " }

  - Existing_Lowercase: { ::f::h::b: " ::g::a ::a a ::e::d " }
  - Existing_Lowernumber: { :a6:a8:a2: " :a7:a1 :a1 a :a5:a4 " }
  - Existing_Caps: { ::F::H::B: " ::G::A ::A a ::E::D " }
  - Existing_Uppernumber: { :A6:A8:A2: " :A7:A1 :A1 a :A5:A4 " }
  - Existing_Numbers: { :6:8:2: " :7:1 :1 a :5:4 " }
  - Existing_Unused: { :˜:œ:ˆ: " :š:ƒ :ƒ a :Ž:Œ " }
  - Unused: { ˜œˆ: " šƒ ƒ a ŽŒ " }
- Two-Word N-Gram
  - Untransformed: { "Tag1": " This is a test " }
  - Existing_Lowercase: ::e " ::c ::b ::a
  - Existing_Lowernumber: :a5 " :a3 :a2 :a1
  - Existing_Caps: ::E " ::C ::B ::A
  - Existing_Uppernumber: :A5 " :A3 :A2 :A1
  - Existing_Numbers: :5 " :3 :2 :1
  - Existing_Unused: :Ž " :Š :ˆ :ƒ
  - Unused: Ž " Š ˆ ƒ

Collision Handling

The need for a collision handling mechanism emerged from the use of existing characters in token replacements to ensure integrity on detransformation. A simple method was adopted to accommodate this. Firstly, prefixes were used in the capital conversion, character and word n-gram data transforms to avoid conflicts between the range of alphabetic, alphanumeric and numeric characters present in both the datasets and the replacement tokens. Secondly, in data transform variations that used existing characters as prefixes, as can be seen in Table 4. These existing characters were doubled up to avoid further collisions.

Table 4. XML and JSON Data Transform Grammar

Transform	Transform Variation	XML Data Transform Variation Grammar	JSON Data Transform Variation Grammar
Tag	Lowercase	<a-z> </a-z> … <a_n-z> </a_n-z>	"a-z": … "a_n-z":
Tag	LowerNumbers	<a1-a_n> </a1-a_n>	"a1-a_n":
Tag	LowerUpper	<a-z> </a-z> … <aA_n-zZ> </aA_n-zZ>	"a-z": … "aA_n-zZ":
Tag	Uppercase	<A-Z> </A-Z> … <A_n-Z> </A_n-Z>	"A-Z": … "A_n-Z":
Tag	UpperLower	<A-Z> </A-Z> … <Aa_n-Zz> </Aa_n-Zz>	"A-Z": … "Aa_n-Zz":
Tag	UpperNumbers	<A1-A_n> </A1-A_n>	"A1-A_n":
Capital	UnusedPrefixes	|a … |z	|a … |z
Capital	ReusedPrefixes	//a … //z	:"a … :"z
Char N-Gram	Existing_Lowercase	<<a … <<z	::a … ::z
Char N-Gram	Existing_Lowernumber	<a0 … <e_n ... <i6	:a0 … :e_n ... :i6
Char N-Gram	Existing_Caps	<<A … <<Z	::A … ::Z
Char N-Gram	Existing_Uppernumber	<A0 … <E_n ... <I6	:A0 … :E_n ... :I6
Char N-Gram	Existing_Numbers	<1 … <26	:1 … :26
Char N-Gram	Existing_Unused	<ƒ, <ˆ, <Š, <Œ, <Ž, <˜, <š, <œ, <ž, <Ÿ, <¡, <¢, <£, <¤, <¥, <¦, <§, <¨, <©, <ª, <«, <¬, <®, <¯, <°, <±	:ƒ, :ˆ, :Š, :Œ, :Ž, :˜, :š, :œ, :ž, :Ÿ, :¡, :¢, :£, :¤, :¥, :¦, :§, :¨, :©, :ª, :«, :¬, :®, :¯, :°, :±
Char N-Gram	Unused	ƒ, ˆ, Š, Œ, Ž, ˜, š, œ, ž, Ÿ, ¡, ¢, £, ¤, ¥, ¦, §, ¨, ©, ª, «, ¬, ®, ¯, °, ±	ƒ, ˆ, Š, Œ, Ž, ˜, š, œ, ž, Ÿ, ¡, ¢, £, ¤, ¥, ¦, §, ¨, ©, ª, «, ¬, ®, ¯, °, ±
Word N-Gram	Existing_Lowercase	<<a … <<z	::a … ::z
Word N-Gram	Existing_Lowernumber	<a0 … <e_n ... <i6	:a0 … :e_n ... :i6

Table 4. XML and JSON Data Transform Grammar

Transform	Transform Variation	XML Data Transform Variation Grammar	JSON Data Transform Variation Grammar
Word N-Gram	Existing_Caps	<<A … <<Z	::A … ::Z
Word N-Gram	Existing_Uppernumber	<A0 … <E$_n$ ... <I6	:A0 … :E$_n$ ... :I6
Word N-Gram	Existing_Numbers	<1 … <26	:1 … :26
Word N-Gram	Existing_Unused	<ƒ, <ˆ, <Š, <Œ, <Ž, <˜, <š, <œ, <ž, <Ÿ, <¡, <¢, <£, <¤, <¥, <¦, <§, <¨, <©, <ª, <«, <¬, <®, <¯, <°, <±	:ƒ, :ˆ, :Š, :Œ, :Ž, :˜, :š, :œ, :ž, :Ÿ, :¡, :¢, :£, :¤, :¥, :¦, :§, :¨, :©, :ª, :«, :¬, :®, :¯, :°, :±
Word N-Gram	Unused	ƒ, ˆ, Š, Œ, Ž, ˜, š, œ, ž, Ÿ, ¡, ¢, £, ¤, ¥, ¦, §, ¨, ©, ª, «, ¬, ®, ¯, °, ±	ƒ, ˆ, Š, Œ, Ž, ˜, š, œ, ž, Ÿ, ¡, ¢, £, ¤, ¥, ¦, §, ¨, ©, ª, «, ¬, ®, ¯, °, ±

3.2.10 Data Transform System Architecture

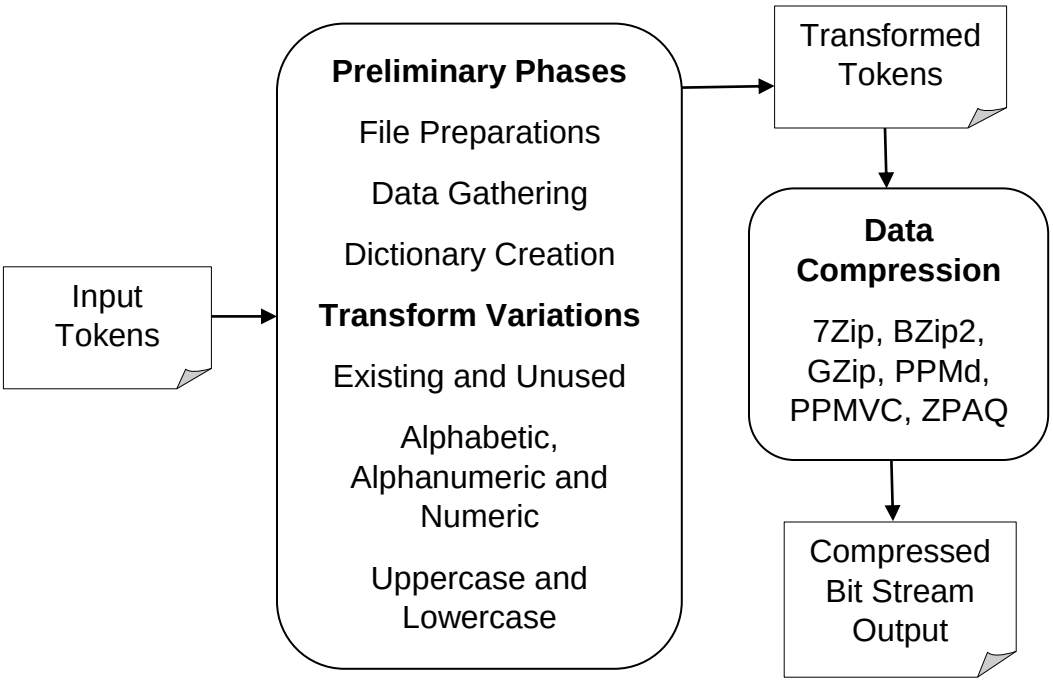

Figure 3. Data Transform System Architecture extended from [67]

Figure 3 shows the data transform system architecture that illustrates the data transform and compression processes involved in this study. It was implemented using Java on Windows 8 operating systems. The use of Java and Windows was due to personal preference and familiarity with this software and operating system. In order to handle memory issues with large files, 64-bit Java was used and the heap space was increased to 4GB using the parameter -Xmx4g. The following describes the data transform and compression process used in this study:

1. All input files undergo the preliminary transform phases:

    1.1. Character encoding - UTF8 without BOM is applied to all files.

    1.2. Distinct XML tag name information is collected for the tag conversion data transform.

1.3. One character n-gram statistics are gathered for capital conversion, character n-gram and word n-gram prefixes to be used in the corresponding dictionaries.

1.4. Two to ten character n-gram statistics are gathered for the character n-gram data transform.

1.5. EOL symbols are converted to spaces to enable accuracy in the word n-gram statistical gathering phase.

1.6. Spaces are inserted after tags to allow for accuracy in the word n-gram statistical gathering phase.

1.7. One to five word n-gram statistics are gathered for the word n gram data transform.

2. The information obtained from the preliminary phases are used to create static dictionaries for all data transform variations.

3. XML and JSON datasets are then transformed using these dictionaries, with data transform variations that are a combination of existing and unused symbols, alphabetical, alphanumeric and numeric characters, and uppercase and lowercase letters.

4. The transformed files then undergo compression using general-purpose data compressors.

A dictionary-based approach was selected as suitable to handle the data transforms used in this study. To take advantage of the information collected during the preliminary phases, static dictionaries were used. Whilst semi-static and adaptive dictionaries could be used in the future, certain decision-making tasks would not be as simple with semi-static and adaptive dictionaries. For example, decisions relating to which prefixes could be used in the dictionaries as well as other collision handling decisions can be fully made certain when information such as the characters and frequency of characters are known to the user prior to the

creation of dictionaries. Whilst this is still possible with semi-dynamic and adaptive dictionaries, they would require an extra pass over the documents during the process and there is still a potential risk of collision handling issues. In this study, static dictionaries were chosen and did not require an extra pass during the actual transform process, as this was carried out during the preliminary transform phases.

Java Programs

The preliminary data transform phases described in step one were conducted using the tools and techniques specified in Section 3.2.7. The dictionaries noted in step two were created manually including ensuring any characters that needed escaping in the character and word n-gram data transform dictionaries.

Steps three and four mentioned above, were executed in Java. Several Java files were created per data format, data group, data transform, transform variation and compression metric, these files were run manually. Whilst the process of detransformation was not included in this study to maintain the focus of this research on data compression over transformed data, detransformation was carried out on a small number of files to ensure that the original data could be reproduced. The following explains how both the XML and JSON Java programs were written and how the data transforms and compression algorithms were used:

1. Data Transforms – All data transforms were run with their pre-prepared dictionaries using HashMap.

2. Compression Ratios – All of the compression algorithms used in this research, XMill, 7Zip, BZip2, GZip, PPMd, PPMVC and ZPAQ, were run as a process over all transformed files using their default compressor settings. The files were compressed and the compression ratios were then computed for all files and the results written to a file.

3. Compression Times – The transformed files were compressed five times. Compression times were then computed for all transformed files and both the average and breakdown of the results were written to files.

4. Decompression Times – The compressed files were decompressed five times. Decompression times were then computed for all compressed files and both the average and breakdown of the results were written to files.

3.2.11 Experimental Assessment

Testing Environment

To ensure robustness, all experiments were conducted on the same testing environment.

Compression Metrics

The metrics used were compression ratios, average compression times and average decompression times, to provide information regarding compressed output size costs and processing times, respectively. These metrics are common in data compression research. However, these metrics measure compressed output size and processing times separately, which has resulted in other researchers combining these metrics in an effort to analyse combined compression efficiency [68], [10], [66], [49]. Whilst a combined compression efficiency metric would be useful in this study to determine, for example, which data transform provided the best compression efficiency in a particular situation, this is dependent on developer requirements. Thus the development of a combined metric would need to be specifically based on the compressed output size and processing user requirements. Therefore, the use of individual compression metrics in this study provides enough information for developers, industry-wide, to be able to use this information to handle data compressed output size and processing according to their needs.

Benchmarks

Providing comparisons with XML-Specific compressors or data transforms, such as XMill [5], [9], [14], [46], [7], [45] or XWRT [71], [70], [72] for example, were both considered for this study, however, only the XMill experiments ran successfully whereas the XWRT experiments were unsuccessful since the program hung due to memory issues. Whilst XML-specific techniques would enable a comparison between data transforms used over XML data and XML-Specific techniques used over untransformed data, they would not provide a comparison between data transforms applied over both XML and JSON. To ensure this study remained focused on using data transforms that can be applied to any data type and format, such as, XML, JSON and others, and that could be used with general-purpose compression techniques, WRT [68] was selected as a benchmark to provide a direct comparison between XML and JSON.

Therefore, the transforms developed in the XML study were compared to the XMill back-end compression algorithms BZip2, GZip and PPMdi and WRT data transform technique optimised for BWT, LZ77, PAQ and PPM compression. The transforms developed in both the JSON and SMILES studies were compared to WRT. However, the XMill results were excluded in order to allow for the direct comparison between XML and JSON. XMill was selected to allow for a comparison for XML data transforms, it was easily available and also because it used similar back-end compressors to those used in WRT. Although many of the data transforms used in this study were also included in the WRT data transform technique, WRT does not include tag conversions, and it also does not use the same data transform variations used in this study. WRT was also selected as it can be applied to any type of data. The transforms were also compared with untransformed data.

Full details of the testing environment used, compression metrics and framework can be found in Section 5.3.

3.2.12 Results Analysis

Various different statistical analysis packages exist to assist in results analysis, namely SPSS [74], MATLAB [48], Microsoft Excel, to name a few. Excel was chosen to analyse the main results in this study and SPSS was chosen for the statistical analysis. The reasons for using Excel and SPSS were due to personal preference and their ease of use. However, with the large amount of data being analysed, SPSS or MATLAB would certainly have been faster in processing the data than Excel.

The results from the data transforms developed in the XML study were compared to XMill, WRT and untransformed data. The results from the JSON study were compared to WRT and untransformed data. Averages were used to summarise the data, data format and compression metric as follows:

- Data transforms developed in this study, XMill, WRT and untransformed comparisons:
  - XML – Percentage that the transforms in this study were better overall compared to XMill, WRT and untransformed data.
  - XML – Percentage that the transform variations in this study were better overall compared to XMill, WRT and untransformed data.
  - JSON – Percentage that the transforms in this study were better overall compared to WRT and untransformed data.
  - JSON – Percentage that the transform variations in this study were better overall compared to WRT and untransformed data.
  - Overall XML and JSON results were also compared with each other with the XMill results excluded from this comparison.

3.3 Case Study

3.3.1 SMILES Case Study

Research Study Observations

The conclusions derived from the literature review conducted in the previous chapter, and the methodology described for the main study (refer to Section 3.2), highlighted the need to also investigate data compression within a data-specific domain as a case study. A case study approach to this part of the investigation would enable investigations on the extent of which data compression can be improved by applying both the data transforms used in the main study (refer to Section 3.2) and data-specific data transforms within a data-specific domain. This was based on the observations from the literature review and the main study:

- The technique to group similar items together based on the information contained in the structural part of an XML document has been used in existing XML-specific compression techniques, such as XMill [5], [9], [14], [46], [7], [45].

- A diverse set of datasets were used in the XML and JSON main study, which included Toxicology datasets including DSSTox, ExpoCast, QSAR and ToxCast – refer to [67] to see where they can be obtained from. On further inspection of the data contained within these datasets, chemical linear notations became of interest.

- With the variety of different chemical molecular structure representations available and the growth of chemical databases, the need to further improve the storage and processing of these molecular representations is of key importance to enable Chemoinformatics practitioners to use this data efficiently. The most widely used chemical linear notation is SMILES [42],

[82], [23], [67].

- As well as being a popular chemical linear notation, the following rationale was used for the selection of SMILES as a suitable and interesting area for this case study:

  - Vocabulary – SMILES contains a simple vocabulary.
  - Character Variation – The variety of alphanumeric characters in SMILES strings make it a good candidate for data-specific transforms for this chemical linear notation.
  - Size – SMILES strings vary in length.

- As with XML and JSON data, general-purpose compressors can be used to compress this type of data to a certain extent, however, these techniques are not specific to SMILES data.

- Some of the data transforms mentioned in the main study (refer to Section 3.2) have not been applied to domain-specific data, particularly SMILES data.

SMILES Case Study

To extend the notion of grouping similar items together according to the document structure [5], [9], [14], [46], [7], [45] a case study approach was set up investigating how much using transforms and transform variations, over a particular similar group of data, would improve the compression of this group of data. The SMILES case study conducted in this thesis extends the study originally published in [67] with additional data transforms, and also uses SMILES data from the same data sources stated in the paper. Further details of the SMILES testing corpus can be found in Section 5.2.3. In order to provide efficient compression of SMILES data, the following formed the basis of the case study conducted in this thesis in order to answer the research questions highlighted in Section 1.4:

- To review the general-purpose data transforms that were used in the main study, and select the relevant data transforms that can be applied specifically to SMILES data.

- To develop data-specific data transforms tailored specifically for SMILES data, that can easily be developed and implemented by developers, and be flexible for transferability so that they can potentially be easily adaptable and tailored to other types of data.

- To investigate data transform variations that can be used alongside these general-purpose and SMILES-specific data transforms.

- To use widely available general-purpose compression algorithms over the transformed data.

- On a practical level, as with the main XML and JSON study, to analyse all results to allow practitioners in the Chemoinformatics domain to make more informed choices on the best data transform, data transform variation and data compression algorithm to use on SMILES, based on their requirements, including, better compressed output size costs, better processing times and a balance of both compressed output size and processing costs to provide information relating to computing resource requirements for their selections.

3.3.2 Data Transforms

Relevant data transforms, previously proposed for use over XML and JSON data in the main study in Section 3.2.3, were selected to be used over SMILES data in this case study. The following decisions were made in the general-purpose data transform selection process:

- Tag Conversions – This data transform was not selected as the SMILES data being investigated was not surrounded by any structural, XML or JSON, tags.

- Capital Conversions – Whilst SMILES strings consist of alphanumeric characters amongst other symbols, this data transform was selected to address the array of atomic symbols used in SMILES data that use uppercase and lowercase letters.

- Character N-Gram Substitution – This data transform was selected purely because it can be used over any type of data, and SMILES strings would potentially provide some interesting character n-grams selections.

- Word N-Gram Substitution – This data transform was not selected as SMILES strings are not separated by spaces and can therefore be considered as one “word”. Whilst frequent one word n-gram substitutions could have been considered, it was felt that this would be best suited for testing a larger amount of SMILES data for best results.

- Space Stuffing and EOL Encoding – These data transforms were also not considered necessary due to the lack of spaces present in SMILES strings, and also because word prediction was not required.

- Punctuation Marks Modelling – This data transform was not selected because it would work better alongside the word n-gram substitution, space stuffing and EOL encoding data transform techniques, that were not selected for this case study. However, it could be used in future work to further improve this study.

As mentioned in the literature review, data-specific transformation techniques have been developed, such as TWRT, which uses words specific to the Java programming language in its dictionary for example [68]. The concept used in TWRT is similar to the data-specific technique developed for transforming SMILES data in the SMILES case study. The SMILES-specific transform is as follows:

Periodic Number

This SMILES-specific data transform was entirely based on substituting atomic elements with their corresponding atomic numbers, with the information sourced from the periodic table. Any numerical character conflicts are resolved by the addition of prefixes prior to atomic element mapping. With the assortment of atomic elements present in SMILES strings, this simple data transform was considered suitable for this type of data [67].

3.3.3 Data Transform Variations

The selection of data transform variation techniques to be used in this case study was based on the same criteria used in the main study (refer to Section 3.2.4). As with the main study, the following data transform variations were used:

- Existing and Unused – In addition to the same concept of using existing characters and unused characters from the main study and also the concepts described in the case study published in [67], as the SMILES data was smaller in size compared to many of the datasets used in the main study, the Star Transform encoding scheme was also used as a data transform variation in this case study.

- Alphabetical, Alphanumeric and Numeric – This data transform variation was included due to the alphanumeric and numeric characters present in SMILES data.

- Uppercase and Lowercase – This data transform variation was suitable to handle the array of uppercase and lowercase characters present in atomic elements within SMILES strings.

3.3.4 General-Purpose Data Compressors

To ensure consistency, the same general-purpose compressors used in the main study (refer to Section 3.2.5), and also used in the case study published in [67], namely 7Zip, BZip2, GZip, PPMd, PPMVC and ZPAQ, were also used in this case study.

3.3.5 Data Transform Properties

It is expected that the data transforms used in the SMILES case study conform to the same properties as stated in the main study and also in the case study published in [67] (refer to Section 3.2.6). In addition to this, this study is also expected to handle any ambiguous SMILES tokens which are mainly present in aromatic atomic elements.

3.3.6 Data Transform Preliminary Phases

The same character n-gram collection data transform preliminary phase, as was conducted in the main study, was also carried out over SMILES data prior to applying data transforms and transform variations (refer to Section 3.2.7).

3.3.7 Data Transform Phases

The following shows the techniques used in the different transform phases of the SMILES case study:

- Capital Conversions – The application of this transform was the same as in the main study (refer to Section 3.2.8). Also, in continuation of the reasons for not preceding words that just contained capital letters with a separate prefix, as mentioned in the main study, it was also not suitable for SMILES data due to its alphanumerical composition and also due to the lack of spaces present in SMILES strings, making each SMILES string a “word” in essence.

- Character N-Gram Transformations – This transform was also applied as per the main study (refer to Section 3.2.8).

- Periodic Number – Numerical characters already present in SMILES strings were prefixed with frequently used existing characters or unused characters, including star encoding, to avoid any conflicts with the next step. Atomic symbols were then substituted with their equivalent atomic numbers from the periodic table to maintain the no ambiguity property. Aromatic elements were transformed in the final stage of this transform also to maintain the no ambiguity property [67].

3.3.8 Data Transform Grammar

The following Table 5 illustrates the grammar applied to SMILES data during the data transform phases. Further descriptions of the data transform variations can be found in Section 5.3. The following are examples of SMILES data and their equivalent transform representations to illustrate the transform scenarios further:

- Caps
  - Untransformed: C1CC(CNCCCCCCC)CCC1CNCCCCCCC
  - UnusedPrefixes: |c1|c|c(|c|n|c|c|c|c|c|c|c)|c|c|c1|c|n|c|c|c|c|c|c
  - ReusedPrefixes: ()c1()c()c(()c()n()c()c()c()c()c()c()c)()c()c()c1()c()n()c()c()c()c()c()c()c
- Two-Char N-gram
  - Untransformed: C1CC(CNCCCCCCC)CCC1CNCCCCCCC
  - Existing_Lowercase: ((f(a((f(((f(bCC((f(f(f(g((f(f(f(a((bCCCCCC
  - Existing_Lowernumber: (l6l(l1((l6l2CC(l6l6l6l7(l6l6l6l1(l2CCCCCC
  - Existing_Caps: ((F(A((F((F(BCC((F(F(F(G((F(F(F(A((BCCCCCC
  - Existing_Uppernumber: (L6L(L1((L6L2CC(L6L6L6L7(L6L6L6L1(L2CCCCCC
  - Existing_Numbers: (6(1((62CC(6667(6661(2CCCCCC
  - Existing_Unused: (˜ƒ(˜(˜˜ˆCC(˜˜˜š(˜˜˜ƒ(ˆCCCCCC
  - Unused: Cƒ˜CˆCCCCCšCCCƒˆCCCCCC
- PeriodicNum
  - Untransformed: C1CC(CNCCCCCCC)CCC1CNCCCCCCC
  - Unused_NumPrefix: ~6|1~6~6(~6~7~6~6~6~6~6~6~6)~6~6~6|1~6~7~6~6~6~6~6~6~6
  - Reused_NumPrefix: ((6()1((6((6(((6((7((6((6((6((6((6((6)((6((6((6()1((6((7((6((6((6((6((6((6((6((6
  - Stars_NumPrefix: *6*)1*6*6(*6*7*6*6*6*6*6*6*6)*6*6*6*)1*6*7*6*6*6*6*6*6*6

Table 5. SMILES Data Transform Grammar

Transform	Transform Variation	SMILES Data Transform Variation Grammar
Capital	UnusedPrefixes	|a … |z
Capital	ReusedPrefixes	()a … ()z
Char N-Gram	Existing_Lowercase	((a … ((z
Char N-Gram	Existing_Lowernumber	(l0 … (r_n ... (a6
Char N-Gram	Existing_Caps	((A … ((Z
Char N-Gram	Existing_Uppernumber	(L0 … (R_n ... (A6
Char N-Gram	Existing_Numbers	(1 … (26
Char N-Gram	Existing_Unused	(ƒ, (ˆ, (Š, (Œ, (Ž, (˜, (š, (œ, (ž, (Ÿ, (¡, (¢, (£, (¤, (¥, (¦, (§, (¨, (©, (ª, («, (¬, (®, (¯, (°, (±
Char N-Gram	Unused	ƒ, ˆ, Š, Œ, Ž, ˜, š, œ, ž, Ÿ, ¡, ¢, £, ¤, ¥, ¦, §, ¨, ©, ª, «, ¬, ®, ¯, °, ±
Periodic Number	Unused NumPrefix	|1-n
Periodic Number	Unused PeriodicNum	;n, ~n, :n, <n, >n
Periodic Number	Reused NumPrefix	)1-n
Periodic Number	Reused PeriodicNum	==n, ((n, (((n, (=)n, (=(n
Periodic Number	Stars NumPrefix	*)1-n
Periodic Number	Stars PeriodicNum	*n, **n, ***n, ****n, *****n

Collision Handling

Similar to the collision handling technique adopted in the main study, prefixes were used in all data transforms used in this case study, to prevent potential issues arising from conflicting characters present in both SMILES data and the replacement tokens. To this effect, a key assumption used in this study was that any aromatic elements present in SMILES strings do not conflict or cause any direct ambiguity with other elements. Using the atomic element Cobalt as an example, the characters that make up its atomic symbol, Co, would therefore not be confused with the atomic symbols for aliphatic Carbon, C, and aromatic Oxygen, O [67].

3.3.9 Data Transform System Architecture

The same system architecture, implementation, approach, data transform and compression procedures used in the main study were also used in this SMILES case study for consistency purposes. Note that regarding the preliminary transform phase procedure, only the character n-gram collection stage applies to this case study. Refer to Section 3.2.10 for further details.

Java Programs

The preliminary data transform phases, dictionary creation and Java programs for the capital conversion and character n-gram data transforms and data compression metrics for SMILES, were as described in Section 3.2.10.

The following explains how the SMILES-Specific periodic number data transform Java programs was written and how the data transform was used:

5. Periodic Number – Regular expressions were used to transform all numeric characters with pre-prepared prefix transform variations. Atomic symbols were then transformed to their corresponding atomic numbers and pre-prepared prefix variations using HashMaps.

3.3.10 Experimental Assessment

To ensure consistency and robustness, all experiments conducted for this case study used the same testing environment, compression metrics and benchmarks as used in the main study, with the exception of the XMill benchmark. Refer to Section 3.2.11 for further details.

3.3.11 Results Analysis

As per the main study (refer to Section 3.2.12) Excel and SPSS were also used to analyse the results in this case study for consistency purposes.

The results from this study were also compared in a similar manner to those in the main study (refer to Section 3.2.12) as can be seen below:

- Data transforms developed in this study, WRT and untransformed comparisons:
  - Percentage that the transforms in this case study were better overall compared to WRT and untransformed data.
  - Percentage that the transform variations in this case study were better overall compared to WRT and untransformed data.

3.4 Chapter Summary

For both the XML and JSON main study and the SMILES case study, this chapter discussed and justified the tools and techniques used to conduct the research for these studies. It also included a discussion on the data transform and compression procedure adopted, experimental assessment and results analysis. The next chapter presents and discusses the results of both studies.

Chapter 4

System Implementation

4.1 Introduction

The previous chapter discussed the methods and techniques used to conduct this research. In particular, it provided a detailed account of the data transforms and transform variations which form the basis of the software described in this chapter to enable developers to carry out experiments on XML, JSON and SMILES data.

This chapter aims to describe the software developed to enable developers to use the transforms developed in this research and also to allow them to further improve it in the future according to their needs.

4.2 XJS Transform and Compression System

This section describes the XML, JSON and SMILES (XJS) Transform and Compression System that has been developed to allow developers to use the data transforms used in this research to transform XML, JSON and SMILES data.

4.2.1 Software Description and Usage

This section both describes how the software was developed and how it can be used to transform and compress files. For consistency purposes, as mentioned in Section 3.2.10, the software was developed using Java and has been tested on both Windows 7 and 8 operating systems. The developed system also adheres to

the same data transformation system architecture shown in Figure 3.

All the Java, class, transform variations, directories, Text2Ngram application and data compressors necessary to provide full functionality of the software are available on the CD. The software can be run by running the `XJSTransform` Java file (refer to the `README` file on the CD for further instructions on how to run the software).

XML Transform Experiments

Figure 4 shows a screenshot of the XML stage of the software.

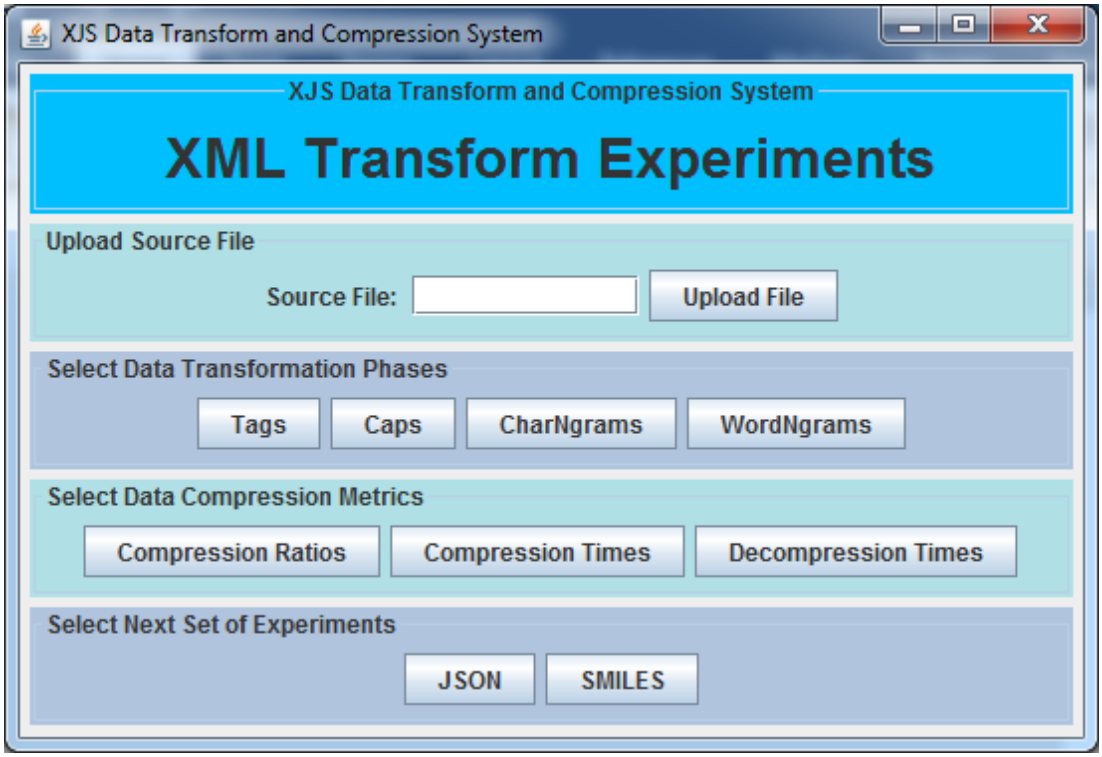

Figure 4. XJS Data Transform and Compression System – XML Transform Experiments

The following steps allow you to successfully transform and compress an XML document:

1. Prior to uploading an XML file, ensure that the prologue and any references to a schema in the document are removed to avoid parsing issues during the XML query phase of the Tags data transform mentioned in step two. Upload an XML source file by clicking on Upload File and navigating to the location of the source file. Once the file has been uploaded, preliminary transforms, such as spaces, are applied to the XML document, to assist with word prediction during the word n-gram collection transform stage.

2. Select the Tags data transform. On selecting this option, the XML document is queried using XQuery to retrieve all distinct tags in the document. The XQuery API for Java (XQJ) (com.saxonica.xqj package) was used in the program. These distinct tags are then used alongside the pre-prepared tag transform variations, refer to Table 4 and the framework in Section 5.3.1 for a description of the relevant transform variations used in the XML set of experiments, to create the dictionaries required for the tag data transforms. The tag data transform is then run using these dictionaries using HashMap.

3. Select the Caps data transform. On selecting this option, the pre-prepared caps transform variations, as mentioned in step two, are used to create the dictionaries required for the caps data transforms. The caps data transform is then run as mentioned in step two.

4. Select the CharNgrams data transform. On selecting this option, the Text2Ngram tool is first run as a process to enable the collection of two to ten character n-grams from the XML document. HashMap is used to sort the character n-grams in descending order and then the top 26 character n-grams are retrieved. Regular expressions are used to ensure certain

characters are escaped to avoid any character handling issues during the transform. The dictionaries are created using the pre-prepared character n-gram transform variations, and the character n-gram data transform is executed as per steps two and three.

5. Select the WordNgram data transform. On selecting this option, the Text2Ngram tool is run as per step four to enable the collection of one to five word n-grams from the XML document. All other steps are as per step three to create the dictionaries and run the word data transform.

6. Select the Compression Ratios metric. On selecting this option, all of the compression algorithms used in this research, 7Zip, BZip2, GZip, PPMd, PPMVC and ZPAQ, are run as a process over all transformed files using their default compressor settings. Compression ratios are then computed for all files and the results are written to the XML\CRatio_Results\ folder.

7. Select the Compression Times metric. On selecting this option, the transformed files are compressed five times as per step six. Compression times are then computed for all transformed files and both the average and breakdown results are written to the XML\ CTime_Results\ folder.

8. Select the Decompression Times metric. On selecting this option, the compressed files are decompressed five times as per steps six and seven. Decompression times are then computed for all compressed files and both the average and breakdown results are written to the XML\DTime_Results\ folder.

9. Select the JSON next set of experiments to run JSON data transforms and compression.

10. Select the SMILES next set of experiments to run SMILES data transforms

and experiments.

JSON Transform Experiments

Figure 5 shows a screenshot of the JSON stage of the software.

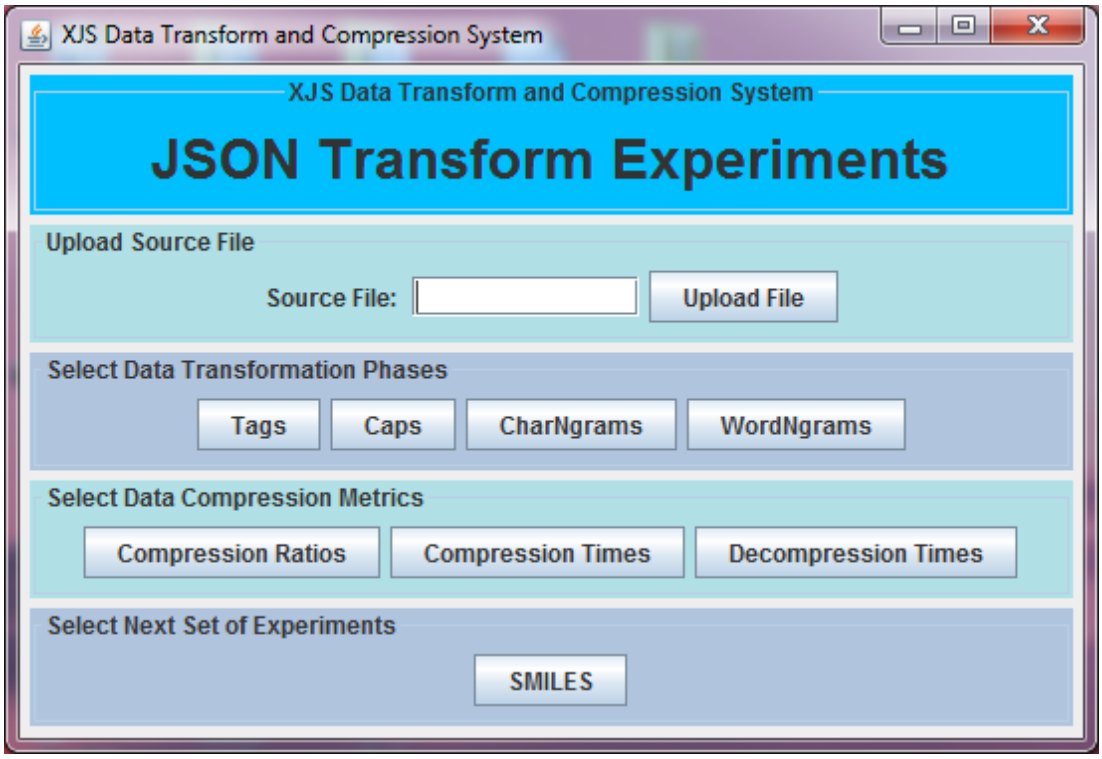

Figure 5. XJS Data Transform and Compression System – JSON Transform Experiments

To successfully transform and compress a JSON document follow the steps stated in the XML Transform Experiments, refer to Table 4 and the framework in Section 5.3.2 for a description of the relevant transform variations used in the JSON set of experiments.

Note that the only difference is during the Tags data transform phase, whereby, unlike in the XML Tag transform phase, the JSON file is not queried for distinct tags. Since these experiments are meant to reflect a comparison between XML and JSON, as per my research, the JSON files used here are assumed to be

the equivalent of the XML files used in the software. Therefore, the same distinct tags are used for this data transform. Future work will involve developing this further to include JSON querying to allow for independent tags to be retrieved if not comparing with XML.

SMILES Transform Experiments

Figure 6 shows a screenshot of the SMILES stage of the software.

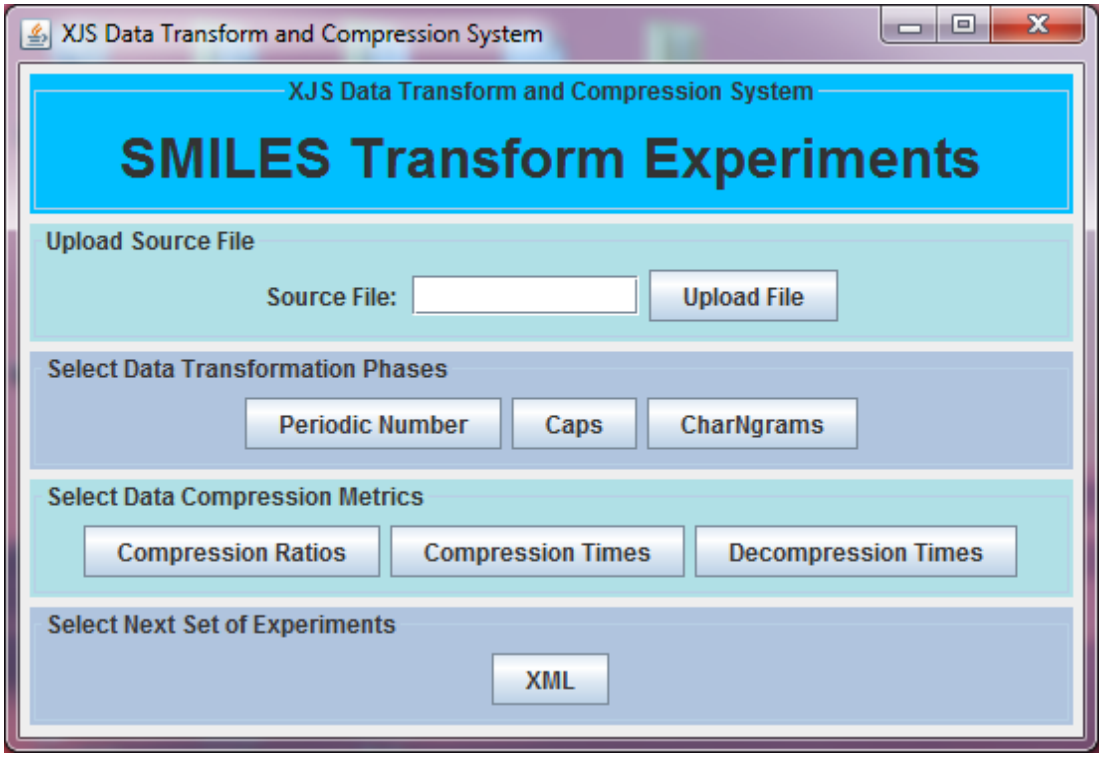

Figure 6. XJS Data Transform and Compression System – SMILES Transform Experiments

To successfully transform and compress SMILES strings contained in a text document follow the steps stated in both the XML and JSON Transform Experiments, refer to Table 5 and the framework in Section 5.3.3 for a description of the relevant transform variations used in the SMILES set of experiments,

Note that the Tags and WordNgrams data transforms are not applicable

here. Another main difference is the addition of the SMILES-Specific Periodic Number data transform. On selecting this data transform regular expressions are used to transform all numeric characters with pre-prepared prefix transform variations. Atomic symbols are then transformed to their corresponding atomic numbers and pre-prepared prefix variations using HashMaps.

4.2.2 Software Improvements

In addition to the future work already mentioned on areas of this software in Section 4.2.1, this section briefly explains how the software can be further enhanced in the future.

- N-gram Collection – This software tool currently uses the Text2Ngram tool to collect both character and word n-grams for the n-gram data transforms. It can be further enhanced by developing n-gram collections in Java to avoid the reliance of an external tool.

- Compression Algorithms – This software tool currently uses external compression algorithms, 7Zip, BZip2, GZip, PPMd, PPMVC and ZPAQ. It can be further enhanced by developing these compression algorithms in Java, again to avoid the reliance of external programs. Compressor settings can also be changed according to your compression requirements.

- Further Extensions – This software can be extended to use other data formats, data transforms, compression algorithms, compression metrics, and so on.

4.3 Chapter Summary

This chapter described the XJS Data Transform and Compression System to allow developers to use the data transforms developed in this thesis in their research. It described how each component in the software worked and demonstrated how the software could be used by developers. It also mentioned how the software can be further improved in the future. The next chapter presents and discusses the results of both the XML and JSON main study and SMILES Case Study described in Chapter 3.

Chapter 5

Results and Discussion

5.1 Introduction

Chapter 3 discussed the methods and techniques used to conduct this research. In particular, it provided a detailed account of the data transforms and transform variations which form the basis of the comparative experiments conducted in this chapter for XML, JSON and the SMILES Case Study.

This chapter presents and discusses the results from these experiments in a detailed and systematic way. To accomplish this, the sections are split across three key areas to accommodate for the XML, JSON, and SMILES Case Study results. The following is presented for the XML, JSON and SMILES experiments:

- Data collection
- Experiment testing environment
- Compression metrics
- Experimental framework
- Result gaps
- Results and analysis
- Results analysis discussion

Finally, a chapter summary is included, stating what has been achieved and the expectations for the next chapter.

5.2 Data Collection

This section describes the data collection processes for XML, JSON and SMILES data.

5.2.1 XML Testing Corpus

XML Datasets

The XML corpus, consisting of 234 files, developed for the experiments was extended from [86] and [85] to include datasets from various other sources. Further details of the datasets used, including dataset characteristics, descriptions and where they can be sourced from, can be found in Table 2 and Appendix A. The datasets chosen covered a range of sizes with the smallest file starting at 774B and the largest file at 390.46MB. Where possible, recent datasets were used from their original sources. Where recent or original datasets were not available, datasets used were taken from [86] and [85].

These files were then transformed according to the techniques discussed in the previous chapter and the resulting transformed files formed the basis of the data used in these experiments. In total, the data corpus for this set of experiments consisted of 26290 files. A breakdown of how these files were allocated per transform is as follows:

- 24423 files for all transforms
  - 1392 files for Tag transforms - 232 files per 6 x transform variations, Tags were not tested for the `cyc` file from the RDF data group and the `treebank` file due to the program hanging during Tag transformations for these files. This problem occurred whilst running the Java program for this transform and was due to memory issues. The Java heap space was increased to 4GB for all experiments, however, this still did not resolve the problems encountered with

these files.

- 468 files for Caps transforms - 234 files per 2 x transform variations.
- 14616 files for Char-Ngram transforms - 2088 files per 7 x transform variations and 232 files per 9 x Char-Ngram levels, Char-Ngrams were not tested for the `enwikiquote` and `enwikiversity` files from the Wikimedia data group due to the program hanging during the Char-Ngram transformations for these files. This problem was again due to the aforementioned memory issues.
- 7947 files for Word-Ngram transforms - 1155 files per 6 x transform variations and 231 files per 5 x Word-Ngram levels, Word-Ngrams were not tested for the `enwikibooks`, `enwikiquote` and `enwikiversity` files from the Wikimedia data group. Also, 1017 files were tested for the Existing_Numbers transform variation due to a problem with Word-Ngram transformations for some of the ToxCast datasets. A list of the affected files can be found in Appendix C. These problems were due to the program hanging during the transformation of these files which was caused by the aforementioned memory issues.

- 936 files for all WRT transforms
  - 234 files for each WRT-BWT, WRT-LZ77, WRT-PAQ and WRT-PPM optimized transform.

- 234 files for all untransformed data.

- 697 files for all XMill experiments
  - 233 files for each XMill-BZip2 and XMill-GZip experiments – XMill was not able to successfully compress the `XBench-TCSD-Small` file from the XBench_XMark data group, so this file was excluded from these experiments. Also, whilst XMill-PPMdi did successfully compress the `enwikibooks` and `enwikiversity` files from the

Wikimedia data group, it was unable to successfully decompress these files due to memory issues. These files were subsequently removed from the XMill-PPMdi experiments, thus testing 231 files for this set of experiments.

Other datasets were considered beneficial for this research and were initially included in the data collection stage but unfortunately had to be excluded later on due to errors produced in the experimental stages due to their large file sizes. An area for future work could incorporate larger files into these experiments using resources with more memory and a better processor. The files affected are listed as follows:

- DBLP (1070MB)
- PSD7003 (786MB)
- enwikisource (5120MB)
- enwiktionary (2420MB)

5.2.2 JSON Testing Corpus

JSON Datasets

The JSON corpus, which was based on the conversion of the XML files described in the previous section, consisted of 231 files. The `e164` file from the RDF data group could not be converted from XML to JSON due to parsing errors that occurred during the conversion. The `enwikibooks` and `enwikiquote` files from the Wikimedia data group could also not be converted from XML to JSON as they exceeded the memory limit. So, these files were excluded from the JSON set of experiments. Further details of the datasets used can be found in Appendix B. These datasets covered a size range from 574B to the largest file at 183.28MB.

As per the XML experiments, these files were also transformed according to the techniques discussed in the previous chapter and the resulting transformed files formed the basis of the data used in these experiments. In total, the data

corpus for this set of experiments consisted of 25463 files. A breakdown of how these files were allocated per transform is as follows:

- 24308 files for all transforms
  - 1374 files for Tag transforms - 229 files per 6 x transform variations, Tags were not tested for the `cyc` file from the RDF data group and `treebank` file due to the program hanging during the Tag transformations for these files. This was due to the memory issues mentioned in Section 5.2.1.
  - 462 files for Caps transforms - 231 files per 2 x transform variations.
  - 14427 files for Char-Ngram transforms - 2061 files per 7 x transform variations and 229 files per 9 x Char-Ngram levels. Char-Ngrams were not tested for `enwikinews` and `enwikiversity` files from the Wikimedia data group due to the program hanging during the Char-Ngram transformations for these files. This was due to the memory issues mentioned in Section 5.2.1.
  - 8045 files for Word-Ngram transforms - 1150 files per 6 x transform variations and 230 files per 5 x Word-Ngram levels. Word-Ngrams were not tested for the `enwikiversity` file from the Wikimedia data group. Also, 1145 files were tested for the Existing_Lowernumber transform variation due to a problem with Word-Ngram transformations for the `enwikinews` file from the Wikimedia data group. These problems were due to the program hanging during the transformation of these files. This was due to the memory issues mentioned in Section 5.2.1.

- 924 files for all WRT transforms
  - 231 files for each WRT-BWT, WRT-LZ77, WRT-PAQ and WRT-PPM optimized transform.

- 231 files for all untransformed data.

5.2.3 SMILES Case Study Testing Corpus

SMILES Datasets

SMILES data was collected from a number of publicly available toxicology datasets and merged into one file of size 1.70MB. Details of where these datasets can be obtained can be found in [67].

The SMILES experiments were extended from [67] to include some of the techniques used in the XML and JSON experiments as well as SMILES-specific data transformations. As per the XML and JSON experiments, these files were also transformed according to the techniques discussed in the previous chapter that could be applied to SMILES data, as well as the use of the developed data-specific techniques that are specific to SMILES data. The resulting transformed files formed the basis of the data used in these experiments. In total, the data corpus for this set of experiments consisted of 73 files. A breakdown of how these files were allocated per transform is as follows:

- 68 files for all transforms
  - 2 files for Caps transforms - 1 file per 2 x transform variations.
  - 63 files for Char-Ngram transforms - 9 files per 7 x transform variations and 1 file per 9 x Char-Ngram levels.
  - 3 files for NumPrefix-PeriodicNum - 1 file per 3 x transform variations.

- 4 files for all WRT transforms
  - 1 file for each WRT-BWT, WRT-LZ77, WRT-PAQ and WRT-PPM optimized transform.

- 1 file for the untransformed data.

5.3 Experiments

This section highlights the testing environment, metrics used and provides details of the experimental framework used for the XML, JSON and SMILES set of experiments.

5.3.1 XML Experiments

Testing Environment

The experiments were run on the following environment:

- Operating System: Windows 8. 1
- Processor: Intel(R) Core(TM) i3-4130 CPU @ 3. 40 GHz
- Installed Memory (RAM): 6. 00GB
- System Type: 64-bit Operating System, x64-based processor

Compression Metrics

The following metrics were computed in Java for these experiments:

- Compression Ratios – The size of the compressed files divided by the size of the uncompressed files in Megabytes (MB).
- Compression Times – The average time taken for each compressor to compress the files in seconds (s).
- Decompression Times – The average time taken for each compressor to decompress the compressed files in seconds (s).

Framework

The experimental framework is as follows:

- The 6 general purpose compressors used on all files except for the XMill experiments were 7Zip [1], BZip2 [15], GZip [75], PPMd [21], PPMVC [34] and ZPAQ [90]. Since, developers of compression techniques provide default settings, it was assumed that these are their recommended settings to be used for most files [66]. To this end, all compressors were used with their default settings. However, the assumption made did not consider the compression of large files such as Wikimedia datasets. A small test was carried out on the `enwikiversity` XML file (184828 KB) whereby GZip was applied to this dataset using compressor setting options –1, –6 (default) and –9. This test showed that the –9 option provided the best compressed output size compared to the other options. This research can be further extended in the future to include the use of non-default compressor settings for further compressed output size and processing analysis.

- The following data transforms and transform variations were tested for all the compression metrics, further details of which can be found in the previous chapter:

  - Tags – Document structure substitution with the following variations:
    - Lowercase – Lowercase alphabetical letters.
    - LowerNumbers – Lowercase alphanumeric symbols.
    - LowerUpper – Both lowercase and uppercase letters of the alphabet, primarily lowercase.
    - Uppercase – Uppercase alphabetical letters.
    - UpperLower – Both uppercase and lowercase letters of the alphabet, primarily uppercase.
    - UpperNumbers – Uppercase alphanumeric symbols.

- Caps – Substitution of uppercase with lowercase letters of the alphabet with the following variations:
  - UnusedPrefixes – Unused characters preceding the substituted letter.
  - ReusedPrefixes – Reused characters preceding the substituted letter.

- Char-Ngrams - N-gram substitution for 2 to 10 characters with the following variations:
  - Existing_Lowercase – Existing characters preceding lowercase alphabetical letters.
  - Existing_Lowernumber – Existing characters preceding lowercase alphanumeric symbols.
  - Existing_Caps – Existing characters preceding uppercase alphabetical letters.
  - Existing_Uppernumber – Existing characters preceding uppercase alphanumeric symbols.
  - Existing_Numbers – Existing characters preceding numeric symbols.
  - Existing_Unused – Existing characters preceding unused symbols.
  - Unused – Unused symbols.

- Word-Ngrams – N-gram substitution for 1 to 5 words with the following variations:
  - Existing_Lowercase – Existing characters preceding lowercase alphabetical letters.
  - Existing_Lowernumber – Existing characters preceding lowercase alphanumeric symbols.
  - Existing_Caps – Existing characters preceding uppercase alphabetical letters.

  - Existing_Uppernumber – Existing characters preceding uppercase alphanumeric symbols.
  - Existing_Numbers – Existing characters preceding numeric symbols.
  - Existing_Unused – Existing characters preceding unused symbols.
  - Unused – Unused symbols.

- Compression metrics were also computed for the following to allow for a comparison of the results:

  - WRT-BWT [34] – WRT transform designed to further enhance BWT compression.
  - WRT-LZ77 [34] – WRT transform designed to further enhance LZ77 compression.
  - WRT-PAQ [34] – WRT transform designed to further enhance PAQ compression.
  - WRT-PPM [34] – WRT transform designed to further enhance PPM compression.
  - Untransformed – Untransformed data.
  - XMill-BZip2 [73] – XMill compression with BZip2.
  - XMill-GZip [73] – XMill compression with GZip.
  - XMill-PPMdi [73] – XMill compression with PPMdi.

- A total of 25593 XML transformed and untransformed files were tested along with a further 697 files for XMill (see Section 5.2.1 for a further breakdown of these files).

- The experiments were conducted on the testing environment described at the beginning of this section.

- Compression and decompression operations were run on each file.

- The average compression and decompression times measured, as described earlier in this section, is based on the average of five executions for each file.

- The total number of runs for the XML set of experiments was 1542550 ((6 * 25593 * 2 * 5) + (697 * 2 * 5)) runs in total.

5.3.2 JSON Experiments

The testing environment, compression metrics and general experimental framework is as per those highlighted in Section 5.3.1. The only differences to the framework relate to the number of files used and the number of experimental runs for the JSON set of experiments; all other conditions remain the same. The changes for JSON are specified below:

- A total of 25463 JSON transformed and untransformed files were tested (see Section 5.2.2 for a further breakdown of these files).

- The total number of runs for the JSON set of experiments per experimental platform was 1527780 (6 * 25463 * 2 * 5), and 3055560 runs in total.

5.3.3 SMILES Experiments

The testing environment, compression metrics and general experimental framework is as per those highlighted in Sections 5.3.1 and 5.3.2, respectively. The differences to the framework relate to the SMILES-specific transforms and transform variations tested, although the Caps and Char-Ngrams transforms used in these experiments are the same as specified in Sections 5.3.1 and 5.3.2,

respectively. Other differences include the number of files used and the number of experimental runs for the SMILES set of experiments; all other conditions remain the same. The changes for SMILES are specified below:

- The following data transforms and transform variations used were tested for all the compression metrics. Further details of which can be found in the previous chapter:

  - Periodic Number – Substitution of atomic symbols to their corresponding atomic number with the following variations:
    - Unused NumPrefix PeriodicNum – Unused characters preceding both the numerical SMILES tokens and also the substituted atomic numbers.
    - Reused NumPrefix PeriodicNum – Reused characters preceding both the numerical SMILES tokens and also the substituted atomic numbers.
    - Stars NumPrefix PeriodicNum – Combined star encoding and existing characters preceding the numerical SMILES tokens, and star encoding scheme entirely used to precede the substituted atomic numbers.

- A total of 73 SMILES transformed and untransformed files were tested (see Section 5.2.3 for a further breakdown of these files).

- The total number of runs for the SMILES set of experiments per experimental platform was 4380 (6 * 73 * 2 * 5), and 8760 runs in total.

5.4 Gaps in the Results

This section highlights the main areas where there are gaps in the results for the XML and JSON set of experiments due to the program hanging for the compressors (for reasons mentioned in Sections 5.2.1 and 5.2.2), data groups and transform variations mentioned below. Other missing data is captured in Section 5.2 and Appendices A, B and C.

5.4.1 XML Result Gaps

The following results for all compression metrics are excluded from this chapter:

- PPMVC compression results for the PSD_SwissProt, RNA and ToxCast data groups.
- PPMd and PPMVC compression results for the Treebank, Wikimedia and XBench_XMark data groups.
- Tags data transform results for the Treebank data group.
- CharNgrams_Unused data transform with PPMd compression and WordNgrams_Unused data transform with PPMd compression for the RNA data group.

5.4.2 JSON Result Gaps

The following results for all compression metrics are excluded from this chapter:

- The same gaps in results mentioned in Section 5.4.1 are also applicable to JSON.
- CharNgrams data transform for the Wikimedia data group.

5.5 XML Empirical Results

The following sections provide the results for the XML experiments implemented on the testing environment, (see Section 5.3 for details of the testing environment used for all experiments). All results analysis was conducted using Microsoft Excel. Refer to Figures 1 to 15 and Tables 1 to 12 in Appendix D for the XML results highlighted in this section.

XML Comparison Compression Ratios

The following shows the data transforms that provided better overall compression ratios compared to WRT, untransformed and XMill experiments. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - Caps (18.26%)
  - CharNgrams (2.25%)
- WRT-LZ77
  - Caps (19.57%)
  - CharNgrams (3.82%)
- WRT-PAQ
  - Caps (23.24%)

  - CharNgrams (8.20%)
- WRT-PPM
  - Caps (22.64%)
  - CharNgrams (7.48%)
- Untransformed
  - Caps (9.29%)
- XMill-BZip2
  - Caps (6.26%)
- XMill-GZip
  - Caps (7.95%)

The following shows the data transform variations that provided better overall compression ratios compared to WRT, untransformed and XMill experiments. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - Caps_ReusedPrefixes (20.11%)
  - Caps_UnusedPrefixes (16.41%)
  - CharNgrams_Existing_Caps (4.62%)
  - CharNgrams_Existing_Lowercase (4.53%)
  - CharNgrams_Existing_UpperNumber (4.49%)
  - CharNgrams_Existing_LowerNumber (4.48%)
  - CharNgrams_Existing_Numbers (2.12%)
  - CharNgrams_Existing_Unused (0.20%)
- WRT-LZ77
  - Caps_ReusedPrefixes (21.40%)
  - Caps_UnusedPrefixes (17.75%)
  - CharNgrams_Existing_Caps (6.15%)
  - CharNgrams_Existing_Lowercase (6.06%)
  - CharNgrams_Existing_UpperNumber (6.02%)
  - CharNgrams_Existing_LowerNumber (6.01%)

    - CharNgrams_Existing_Numbers (3.69%)
    - CharNgrams_Existing_Unused (1.80%)
- WRT-PAQ
    - Caps_ReusedPrefixes (24.98%)
    - Caps_UnusedPrefixes (21.50%)
    - CharNgrams_Existing_Caps (10.43%)
    - CharNgrams_Existing_Lowercase (10.34%)
    - CharNgrams_Existing_UpperNumber (10.30%)
    - CharNgrams_Existing_LowerNumber (10.30%)
    - CharNgrams_Existing_Numbers (8.08%)
    - CharNgrams_Existing_Unused (6.28%)
    - CharNgrams_Unused (1.61%)
    - Tags_UpperNumbers (1.08%)
    - Tags_LowerNumbers (1.07%)
- WRT-PPM
    - Caps_ReusedPrefixes (24.39%)
    - Caps_UnusedPrefixes (20.88%)
    - CharNgrams_Existing_Caps (9.73%)
    - CharNgrams_Existing_Lowercase (9.64%)
    - CharNgrams_Existing_UpperNumber (9.60%)
    - CharNgrams_Existing_LowerNumber (9.59%)
    - CharNgrams_Existing_Numbers (7.36%)
    - CharNgrams_Existing_Unused (5.54%)
    - CharNgrams_Unused (0.84%)
    - Tags_UpperNumbers (0.30%)
    - Tags_LowerNumbers (0.29%)
- Untransformed
    - Caps_ReusedPrefixes (11.35%)
    - Caps_UnusedPrefixes (7.24%)
- XMill-BZip2
    - Caps_ReusedPrefixes (8.38%)

  - Caps_UnusedPrefixes (4.13%)
- XMill-GZip
  - Caps_ReusedPrefixes (10.03%)
  - Caps_UnusedPrefixes (5.86%)

XML Comparison Compression Times

The following shows the data transforms that provided better overall compression times compared to WRT, untransformed and XMill experiments. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - WordNgrams (26.91%)
  - CharNgrams (0.63%)
- WRT-LZ77
  - WordNgrams (23.06%)
- WRT-PAQ
  - WordNgrams (24.77%)
- WRT-PPM
  - WordNgrams (24.83%)
- Untransformed
  - WordNgrams (42.77%)
  - CharNgrams (22.19%)
  - Tags (16.71%)
- XMill-BZip2
  - Tags (13.37%)
  - CharNgrams (19.07%)
  - WordNgrams (40.48%)

The following shows the data transform variations that provided better overall compression times compared to WRT, untransformed and XMill experiments. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - WordNgrams_Unused (29.76%)
  - WordNgrams_Existing_Unused (28.67%)
  - WordNgrams_Existing_Caps (27.60%)
  - WordNgrams_Existing_Lowercase (27.49%)
  - WordNgrams_Existing_UpperNumber (27.34%)
  - WordNgrams_Existing_LowerNumber (25.62%)
  - WordNgrams_Existing_Numbers (21.34%)
  - CharNgrams_Unused (8.32%)
  - CharNgrams_Existing_Unused (2.47%)
  - CharNgrams_Existing_Numbers (0.06%)
- WRT-LZ77
  - WordNgrams_Unused (26.06%)
  - WordNgrams_Existing_Unused (24.91%)
  - WordNgrams_Existing_Caps (23.78%)
  - WordNgrams_Existing_Lowercase (23.67%)
  - WordNgrams_Existing_UpperNumber (23.51%)
  - WordNgrams_Existing_LowerNumber (21.70%)
  - WordNgrams_Existing_Numbers (17.19%)
  - CharNgrams_Unused (3.48%)
- WRT-PAQ
  - WordNgrams_Unused (27.70%)
  - WordNgrams_Existing_Unused (26.58%)
  - WordNgrams_Existing_Caps (25.48%)
  - WordNgrams_Existing_Lowercase (25.37%)
  - WordNgrams_Existing_UpperNumber (25.21%)
  - WordNgrams_Existing_LowerNumber (23.44%)
  - WordNgrams_Existing_Numbers (19.04%)
  - CharNgrams_Unused (5.63%)
- WRT-PPM
  - WordNgrams_Unused (27.76%)

- WordNgrams_Existing_Unused (26.64%)
- WordNgrams_Existing_Caps (25.53%)
- WordNgrams_Existing_Lowercase (25.43%)
- WordNgrams_Existing_UpperNumber (25.27%)
- WordNgrams_Existing_LowerNumber (23.50%)
- WordNgrams_Existing_Numbers (19.10%)
- CharNgrams_Unused (5.70%)

- Untransformed
  - WordNgrams_Unused (45.00%)
  - WordNgrams_Existing_Unused (44.15%)
  - WordNgrams_Existing_Caps (43.31%)
  - WordNgrams_Existing_Lowercase (43.22%)
  - WordNgrams_Existing_UpperNumber (43.10%)
  - WordNgrams_Existing_LowerNumber (41.76%)
  - WordNgrams_Existing_Numbers (38.41%)
  - CharNgrams_Unused (28.21%)
  - CharNgrams_Existing_Unused (23.63%)
  - CharNgrams_Existing_Numbers (21.74%)
  - CharNgrams_Existing_LowerNumber (20.95%)
  - CharNgrams_Existing_Caps (20.38%)
  - CharNgrams_Existing_Lowercase (20.27%)
  - CharNgrams_Existing_UpperNumber (20.19%)
  - Tags_Lowercase (18.28%)
  - Tags_Uppercase (18.28%)
  - Tags_LowerUpper (18.18%)
  - Tags_UpperLower (18.16%)
  - Tags_LowerNumbers (13.77%)
  - Tags_UpperNumbers (13.59%)
- XMill-BZip2
  - Tags_Lowercase (15.01%)
  - Tags_LowerNumbers (10.31%)

- Tags_LowerUpper (14.90%)
- Tags_Uppercase (15.00%)
- Tags_UpperLower (14.88%)
- Tags_UpperNumbers (10.13%)
- CharNgrams_Existing_Caps (17.19%)
- CharNgrams_Existing_Lowercase (17.07%)
- CharNgrams_Existing_LowerNumber (17.79%)
- CharNgrams_Existing_Numbers (18.61%)
- CharNgrams_Existing_Unused (20.57%)
- CharNgrams_Existing_UpperNumber (16.99%)
- CharNgrams_Unused (25.33%)
- WordNgrams_Existing_Caps (41.03%)
- WordNgrams_Existing_Lowercase (40.95%)
- WordNgrams_Existing_LowerNumber (39.43%)
- WordNgrams_Existing_Numbers (35.94%)
- WordNgrams_Existing_Unused (41.91%)
- WordNgrams_Existing_UpperNumber (40.83%)
- WordNgrams_Unused (42.80%)

XML Comparison Decompression Times

The following shows the data transforms that provided better overall decompression times compared to WRT, untransformed and XMill experiments. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - WordNgrams (22.29%)
  - Tags (9.96%)
  - CharNgrams (4.42%)
- WRT-LZ77
  - WordNgrams (21.51%)
  - Tags (9.06%)

  - CharNgrams (3.46%)
- WRT-PAQ
  - WordNgrams (18.54%)
  - Tags (5.61%)
- WRT-PPM
  - WordNgrams (18.65%)
  - Tags (5.74%)
- Untransformed
  - WordNgrams (28.12%)
  - Tags (16.71%)
  - CharNgrams (11.58%)
- XMill-BZip2
  - Tags (41.09%)
  - Caps (14.56%)
  - CharNgrams (37.47%)
  - WordNgrams (49.16%)
- XMill-PPMdi
  - Tags (74.12%)
  - Caps (62.47%)
  - CharNgrams (72.53%)
  - WordNgrams (77.67%)

The following shows the data transform variations that provided better overall decompression times compared to WRT, untransformed and XMill experiments. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - WordNgrams_Unused (28.34%)
  - WordNgrams_Existing_Unused (26.03%)
  - WordNgrams_Existing_Lowercase (25.50%)
  - WordNgrams_Existing_LowerNumber (22.90%)

  - WordNgrams_Existing_UpperNumber (21.66%)
  - WordNgrams_Existing_Numbers (18.54%)
  - WordNgrams_Existing_Caps (12.74%)
  - CharNgrams_Unused (11.85%)
  - Tags_LowerUpper (11.69%)
  - Tags_UpperLower (11.23%)
  - Tags_Uppercase (11.19%)
  - Tags_Lowercase (10.95%)
  - Tags_UpperNumbers (7.81%)
  - Tags_LowerNumbers (6.88%)
  - CharNgrams_Existing_Unused (6.58%)
  - CharNgrams_Existing_Numbers (5.00%)
  - CharNgrams_Existing_Caps (2.42%)
  - CharNgrams_Existing_LowerNumber (2.03%)
  - CharNgrams_Existing_Lowercase (1.72%)
  - CharNgrams_Existing_UpperNumber (1.41%)
- WRT-LZ77
  - WordNgrams_Unused (27.62%)
  - WordNgrams_Existing_Unused (25.29%)
  - WordNgrams_Existing_Lowercase (24.75%)
  - WordNgrams_Existing_LowerNumber (22.13%)
  - WordNgrams_Existing_UpperNumber (20.87%)
  - WordNgrams_Existing_Numbers (17.72%)
  - WordNgrams_Existing_Caps (11.86%)
  - CharNgrams_Unused (10.96%)
  - Tags_LowerUpper (10.81%)
  - Tags_UpperLower (10.34%)
  - Tags_Uppercase (10.30%)
  - Tags_Lowercase (10.06%)
  - Tags_UpperNumbers (6.88%)
  - Tags_LowerNumbers (5.94%)

  - CharNgrams_Existing_Unused (5.64%)
  - CharNgrams_Existing_Numbers (4.04%)
  - CharNgrams_Existing_Caps (1.44%)
  - CharNgrams_Existing_LowerNumber (1.05%)
  - CharNgrams_Existing_Lowercase (0.74%)
  - CharNgrams_Existing_UpperNumber (0.42%)
- WRT-PAQ
  - WordNgrams_Unused (24.88%)
  - WordNgrams_Existing_Unused (22.46%)
  - WordNgrams_Existing_Lowercase (21.90%)
  - WordNgrams_Existing_LowerNumber (19.18%)
  - WordNgrams_Existing_UpperNumber (17.87%)
  - WordNgrams_Existing_Numbers (14.60%)
  - WordNgrams_Existing_Caps (8.52%)
  - CharNgrams_Unused (7.59%)
  - Tags_LowerUpper (7.43%)
  - Tags_UpperLower (6.95%)
  - Tags_Uppercase (6.90%)
  - Tags_Lowercase (6.65%)
  - Tags_UpperNumbers (3.35%)
  - Tags_LowerNumbers (2.38%)
  - CharNgrams_Existing_Unused (2.07%)
  - CharNgrams_Existing_Numbers (0.41%)
- WRT-PPM
  - WordNgrams_Unused (24.98%)
  - WordNgrams_Existing_Unused (22.56%)
  - WordNgrams_Existing_Lowercase (22.01%)
  - WordNgrams_Existing_LowerNumber (19.28%)
  - WordNgrams_Existing_UpperNumber (17.98%)
  - WordNgrams_Existing_Numbers (14.72%)
  - WordNgrams_Existing_Caps (8.64%)

- CharNgrams_Unused (7.71%)
- Tags_LowerUpper (7.55%)
- Tags_UpperLower (7.07%)
- Tags_Uppercase (7.03%)
- Tags_Lowercase (6.78%)
- Tags_UpperNumbers (3.48%)
- Tags_LowerNumbers (2.51%)
- CharNgrams_Existing_Unused (2.20%)
- CharNgrams_Existing_Numbers (0.54%)

■ Untransformed

- WordNgrams_Unused (33.71%)
- WordNgrams_Existing_Unused (31.57%)
- WordNgrams_Existing_Lowercase (31.08%)
- WordNgrams_Existing_LowerNumber (28.68%)
- WordNgrams_Existing_UpperNumber (27.53%)
- WordNgrams_Existing_Numbers (24.64%)
- WordNgrams_Existing_Caps (19.27%)
- CharNgrams_Unused (18.45%)
- Tags_LowerUpper (18.31%)
- Tags_UpperLower (17.88%)
- Tags_Uppercase (17.85%)
- Tags_Lowercase (17.63%)
- Tags_UpperNumbers (14.72%)
- Tags_LowerNumbers (13.85%)
- CharNgrams_Existing_Unused (13.58%)
- CharNgrams_Existing_Numbers (12.11%)
- CharNgrams_Existing_Caps (9.73%)
- CharNgrams_Existing_LowerNumber (9.37%)
- CharNgrams_Existing_Lowercase (9.09%)
- CharNgrams_Existing_UpperNumber (8.80%)

- XMill-BZip2
  - Tags_Lowercase (41.75%)
  - Tags_LowerNumbers (39.08%)
  - Tags_LowerUpper (42.23%)
  - Tags_Uppercase (41.90%)
  - Tags_UpperLower (41.93%)
  - Tags_UpperNumbers (39.69%)
  - Caps_ReusedPrefixes (11.09%)
  - Caps_UnusedPrefixes (18.03%)
  - CharNgrams_Existing_Caps (36.16%)
  - CharNgrams_Existing_Lowercase (35.71%)
  - CharNgrams_Existing_LowerNumber (35.91%)
  - CharNgrams_Existing_Numbers (37.85%)
  - CharNgrams_Existing_Unused (38.88%)
  - CharNgrams_Existing_UpperNumber (35.50%)
  - CharNgrams_Unused (42.33%)
  - WordNgrams_Existing_Caps (42.91%)
  - WordNgrams_Existing_Lowercase (51.26%)
  - WordNgrams_Existing_LowerNumber (49.56%)
  - WordNgrams_Existing_Numbers (46.71%)
  - WordNgrams_Existing_Unused (51.61%)
  - WordNgrams_Existing_UpperNumber (48.75%)
  - WordNgrams_Unused (53.12%)
- XMill-PPMdi
  - Tags_Lowercase (74.41%)
  - Tags_LowerNumbers (73.24%)
  - Tags_LowerUpper (74.62%)
  - Tags_Uppercase (74.48%)
  - Tags_UpperLower (74.49%)
  - Tags_UpperNumbers (73.50%)
  - Caps_ReusedPrefixes (60.94%)

- Caps_UnusedPrefixes (63.99%)
- CharNgrams_Existing_Caps (71.96%)
- CharNgrams_Existing_Lowercase (71.76%)
- CharNgrams_Existing_LowerNumber (71.85%)
- CharNgrams_Existing_Numbers (72.70%)
- CharNgrams_Existing_Unused (73.15%)
- CharNgrams_Existing_UpperNumber (71.67%)
- CharNgrams_Unused (74.67%)
- WordNgrams_Existing_Caps (74.92%)
- WordNgrams_Existing_Lowercase (78.59%)
- WordNgrams_Existing_LowerNumber (77.84%)
- WordNgrams_Existing_Numbers (76.59%)
- WordNgrams_Existing_Unused (78.74%)
- WordNgrams_Existing_UpperNumber (77.48%)
- WordNgrams_Unused (79.41%)

5.6 JSON Empirical Results

The following sections demonstrate the results for the JSON experiments with implementation and results analysis the same as in Section 5.5. Refer to Figures 1 to 15 and Tables 13 to 24 in Appendix D for the JSON results highlighted in this section.

JSON Comparison Compression Ratios

The following shows the data transforms that provided better overall compression ratios compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - Caps (18.64%)
  - CharNgrams (1.62%)
- WRT-LZ77
  - Caps (19.68%)
  - CharNgrams (2.89%)
  - Tags (0.20%)
- WRT-PAQ
  - Caps (21.71%)
  - CharNgrams (5.34%)
  - Tags (2.72%)
- WRT-PPM
  - Caps (21.25%)
  - CharNgrams (4.78%)
  - Tags (2.15%)
- Untransformed
  - Caps (9.53%)

The following shows the data transform variations that provided better overall compression ratios compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - Caps_ReusedPrefixes (20.73%)
  - Caps_UnusedPrefixes (16.54%)
  - CharNgrams_Existing_Caps (4.06%)
  - CharNgrams_Existing_Lowercase (3.98%)
  - CharNgrams_Existing_UpperNumber (3.93%)
  - CharNgrams_Existing_LowerNumber (3.93%)
  - CharNgrams_Existing_Numbers (1.36%)
  - Tags_UpperNumbers (0.42%)

    - Tags_LowerNumbers (0.40%)
- WRT-LZ77
    - Caps_ReusedPrefixes (21.75%)
    - Caps_UnusedPrefixes (17.62%)
    - CharNgrams_Existing_Caps (5.30%)
    - CharNgrams_Existing_Lowercase (5.22%)
    - CharNgrams_Existing_UpperNumber (5.17%)
    - CharNgrams_Existing_LowerNumber (5.16%)
    - CharNgrams_Existing_Numbers (2.64%)
    - Tags_UpperNumbers (1.71%)
    - Tags_LowerNumbers (1.69%)
    - CharNgrams_Existing_Unused (0.83%)
- WRT-PAQ
    - Caps_ReusedPrefixes (23.73%)
    - Caps_UnusedPrefixes (19.70%)
    - CharNgrams_Existing_Caps (7.69%)
    - CharNgrams_Existing_Lowercase (7.61%)
    - CharNgrams_Existing_UpperNumber (7.56%)
    - CharNgrams_Existing_LowerNumber (7.56%)
    - CharNgrams_Existing_Numbers (5.10%)
    - Tags_UpperNumbers (4.19%)
    - Tags_LowerNumbers (4.17%)
    - CharNgrams_Existing_Unused (3.34%)
    - Tags_Lowercase (2.03%)
    - Tags_Uppercase (2.02%)
    - Tags_LowerUpper (1.97%)
    - Tags_UpperLower (1.93%)
- WRT-PPM
    - Caps_ReusedPrefixes (23.28%)
    - Caps_UnusedPrefixes (19.23%)
    - CharNgrams_Existing_Caps (7.15%)

  - CharNgrams_Existing_Lowercase (7.07%)
  - CharNgrams_Existing_UpperNumber (7.02%)
  - CharNgrams_Existing_LowerNumber (7.01%)
  - CharNgrams_Existing_Numbers (4.54%)
  - Tags_UpperNumbers (3.62%)
  - Tags_LowerNumbers (3.60%)
  - CharNgrams_Existing_Unused (2.77%)
  - Tags_Lowercase (1.45%)
  - Tags_Uppercase (1.44%)
  - Tags_LowerUpper (1.39%)
  - Tags_UpperLower (1.35%)
- Untransformed
  - Caps_ReusedPrefixes (11.86%)
  - Caps_UnusedPrefixes (7.21%)

JSON Comparison Compression Times

The following shows the data transforms that provided better overall compression times compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - CharNgrams (12.16%)
  - WordNgrams (10.00%)
- WRT-LZ77
  - CharNgrams (9.88%)
  - WordNgrams (7.66%)
- WRT-PAQ
  - CharNgrams (12.37%)
  - WordNgrams (10.22%)
- WRT-PPM
  - CharNgrams (12.41%)

  - WordNgrams (10.26%)
- Untransformed
  - CharNgrams (25.32%)
  - WordNgrams (23.49%)
  - Tags (14.65%)

The following shows the data transform variations that provided better overall compression times compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - CharNgrams_Unused (18.07%)
  - WordNgrams_Existing_LowerNumber (16.96%)
  - CharNgrams_Existing_Unused (13.70%)
  - CharNgrams_Existing_Numbers (12.55%)
  - CharNgrams_Existing_LowerNumber (10.42%)
  - CharNgrams_Existing_UpperNumber (10.19%)
  - CharNgrams_Existing_Lowercase (10.12%)
  - CharNgrams_Existing_Caps (10.11%)
  - WordNgrams_Unused (9.96%)
  - WordNgrams_Existing_Unused (9.63%)
  - WordNgrams_Existing_Numbers (8.56%)
  - WordNgrams_Existing_Caps (8.41%)
  - WordNgrams_Existing_UpperNumber (8.40%)
  - WordNgrams_Existing_Lowercase (8.11%)
  - Tags_Uppercase (0.43%)
  - Tags_UpperLower (0.41%)
  - Tags_LowerUpper (0.22%)
  - Tags_Lowercase (0.14%)

- WRT-LZ77
  - CharNgrams_Unused (15.94%)
  - WordNgrams_Existing_LowerNumber (14.81%)
  - CharNgrams_Existing_Unused (11.45%)
  - CharNgrams_Existing_Numbers (10.28%)
  - CharNgrams_Existing_LowerNumber (8.09%)
  - CharNgrams_Existing_UpperNumber (7.86%)
  - CharNgrams_Existing_Lowercase (7.78%)
  - CharNgrams_Existing_Caps (7.78%)
  - WordNgrams_Unused (7.62%)
  - WordNgrams_Existing_Unused (7.28%)
  - WordNgrams_Existing_Numbers (6.18%)
  - WordNgrams_Existing_Caps (6.03%)
  - WordNgrams_Existing_UpperNumber (6.02%)
  - WordNgrams_Existing_Lowercase (5.73%)
- WRT-PAQ
  - CharNgrams_Unused (18.27%)
  - WordNgrams_Existing_LowerNumber (17.17%)
  - CharNgrams_Existing_Unused (13.91%)
  - CharNgrams_Existing_Numbers (12.76%)
  - CharNgrams_Existing_LowerNumber (10.64%)
  - CharNgrams_Existing_UpperNumber (10.41%)
  - CharNgrams_Existing_Lowercase (10.34%)
  - CharNgrams_Existing_Caps (10.33%)
  - WordNgrams_Unused (10.18%)
  - WordNgrams_Existing_Unused (9.85%)
  - WordNgrams_Existing_Numbers (8.78%)
  - WordNgrams_Existing_Caps (8.63%)
  - WordNgrams_Existing_UpperNumber (8.63%)
  - WordNgrams_Existing_Lowercase (8.34%)
  - Tags_Uppercase (0.67%)

    - Tags_UpperLower (0.66%)
    - Tags_LowerUpper (0.47%)
    - Tags_Lowercase (0.39%)
- WRT-PPM
    - CharNgrams_Unused (18.31%)
    - WordNgrams_Existing_LowerNumber (17.20%)
    - CharNgrams_Existing_Unused (13.95%)
    - CharNgrams_Existing_Numbers (12.80%)
    - CharNgrams_Existing_LowerNumber (10.68%)
    - CharNgrams_Existing_UpperNumber (10.45%)
    - CharNgrams_Existing_Lowercase (10.38%)
    - CharNgrams_Existing_Caps (10.37%)
    - WordNgrams_Unused (10.22%)
    - WordNgrams_Existing_Unused (9.89%)
    - WordNgrams_Existing_Numbers (8.82%)
    - WordNgrams_Existing_Caps (8.67%)
    - WordNgrams_Existing_UpperNumber (8.67%)
    - WordNgrams_Existing_Lowercase (8.38%)
    - Tags_Uppercase (0.71%)
    - Tags_UpperLower (0.70%)
    - Tags_LowerUpper (0.51%)
    - Tags_Lowercase (0.43%)
- Untransformed
    - CharNgrams_Unused (30.35%)
    - WordNgrams_Existing_LowerNumber (29.41%)
    - CharNgrams_Existing_Unused (26.63%)
    - CharNgrams_Existing_Numbers (25.65%)
    - CharNgrams_Existing_LowerNumber (23.84%)
    - CharNgrams_Existing_UpperNumber (23.65%)
    - CharNgrams_Existing_Lowercase (23.59%)
    - CharNgrams_Existing_Caps (23.58%)

  - WordNgrams_Unused (23.45%)
  - WordNgrams_Existing_Unused (23.17%)
  - WordNgrams_Existing_Numbers (22.26%)
  - WordNgrams_Existing_Caps (22.13%)
  - WordNgrams_Existing_UpperNumber (22.13%)
  - WordNgrams_Existing_Lowercase (21.88%)
  - Tags_Uppercase (15.35%)
  - Tags_UpperLower (15.33%)
  - Tags_LowerUpper (15.17%)
  - Tags_Lowercase (15.11%)
  - Tags_LowerNumbers (13.49%)
  - Tags_UpperNumbers (13.43%)

JSON Comparison Decompression Times

The following shows the data transforms that provided better overall decompression times compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - WordNgrams (18.30%)
  - Tags (17.66%)
  - CharNgrams (15.77%)
- WRT-LZ77
  - WordNgrams (17.56%)
  - Tags (16.92%)
  - CharNgrams (15.00%)
- WRT-PAQ
  - WordNgrams (17.62%)
  - Tags (16.98%)
  - CharNgrams (15.07%)

- WRT-PPM
  - WordNgrams (17.89%)
  - Tags (17.25%)
  - CharNgrams (15.35%)
- Untransformed
  - WordNgrams (26.87%)
  - Tags (26.30%)
  - CharNgrams (24.60%)

The following shows the data transform variations that provided better overall decompression times compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - CharNgrams_Unused (22.53%)
  - WordNgrams_Unused (20.90%)
  - WordNgrams_Existing_LowerNumber (20.83%)
  - Tags_UpperLower (18.85%)
  - Tags_Lowercase (18.49%)
  - Tags_LowerUpper (18.34%)
  - WordNgrams_Existing_Unused (18.26%)
  - Tags_Uppercase (17.95%)
  - CharNgrams_Existing_Unused (17.50%)
  - WordNgrams_Existing_Numbers (17.34%)
  - WordNgrams_Existing_UpperNumber (17.22%)
  - WordNgrams_Existing_Lowercase (16.92%)
  - WordNgrams_Existing_Caps (16.65%)
  - Tags_UpperNumbers (16.58%)
  - CharNgrams_Existing_Numbers (15.81%)
  - Tags_LowerNumbers (15.76%)
  - CharNgrams_Existing_UpperNumber (13.77%)

  - CharNgrams_Existing_Caps (13.64%)
  - CharNgrams_Existing_Lowercase (13.60%)
  - CharNgrams_Existing_LowerNumber (13.59%)
- WRT-LZ77
  - CharNgrams_Unused (21.82%)
  - WordNgrams_Unused (20.18%)
  - WordNgrams_Existing_LowerNumber (20.12%)
  - Tags_UpperLower (18.12%)
  - Tags_Lowercase (17.75%)
  - Tags_LowerUpper (17.60%)
  - WordNgrams_Existing_Unused (17.52%)
  - Tags_Uppercase (17.21%)
  - CharNgrams_Existing_Unused (16.76%)
  - WordNgrams_Existing_Numbers (16.59%)
  - WordNgrams_Existing_UpperNumber (16.46%)
  - WordNgrams_Existing_Lowercase (16.17%)
  - WordNgrams_Existing_Caps (15.89%)
  - Tags_UpperNumbers (15.82%)
  - CharNgrams_Existing_Numbers (15.04%)
  - Tags_LowerNumbers (14.99%)
  - CharNgrams_Existing_UpperNumber (12.99%)
  - CharNgrams_Existing_Caps (12.86%)
  - CharNgrams_Existing_Lowercase (12.81%)
  - CharNgrams_Existing_LowerNumber (12.81%)
- WRT-PAQ
  - CharNgrams_Unused (21.88%)
  - WordNgrams_Unused (20.25%)
  - WordNgrams_Existing_LowerNumber (20.18%)
  - Tags_UpperLower (18.18%)
  - Tags_Lowercase (17.82%)
  - Tags_LowerUpper (17.66%)

  - WordNgrams_Existing_Unused (17.58%)
  - Tags_Uppercase (17.27%)
  - CharNgrams_Existing_Unused (16.82%)
  - WordNgrams_Existing_Numbers (16.65%)
  - WordNgrams_Existing_UpperNumber (16.53%)
  - WordNgrams_Existing_Lowercase (16.23%)
  - WordNgrams_Existing_Caps (15.96%)
  - Tags_UpperNumbers (15.89%)
  - CharNgrams_Existing_Numbers (15.11%)
  - Tags_LowerNumbers (15.06%)
  - CharNgrams_Existing_UpperNumber (13.06%)
  - CharNgrams_Existing_Caps (12.92%)
  - CharNgrams_Existing_Lowercase (12.88%)
  - CharNgrams_Existing_LowerNumber (12.88%)
- WRT-PPM
  - CharNgrams_Unused (22.14%)
  - WordNgrams_Unused (20.51%)
  - WordNgrams_Existing_LowerNumber (20.44%)
  - Tags_UpperLower (18.45%)
  - Tags_Lowercase (18.08%)
  - Tags_LowerUpper (17.93%)
  - WordNgrams_Existing_Unused (17.85%)
  - Tags_Uppercase (17.54%)
  - CharNgrams_Existing_Unused (17.09%)
  - WordNgrams_Existing_Numbers (16.92%)
  - WordNgrams_Existing_UpperNumber (16.80%)
  - WordNgrams_Existing_Lowercase (16.51%)
  - WordNgrams_Existing_Caps (16.23%)
  - Tags_UpperNumbers (16.17%)
  - CharNgrams_Existing_Numbers (15.39%)
  - Tags_LowerNumbers (15.34%)

- CharNgrams_Existing_UpperNumber (13.34%)
- CharNgrams_Existing_Caps (13.21%)
- CharNgrams_Existing_Lowercase (13.17%)
- CharNgrams_Existing_LowerNumber (13.16%)

- Untransformed
  - CharNgrams_Unused (30.65%)
  - WordNgrams_Unused (29.20%)
  - WordNgrams_Existing_LowerNumber (29.14%)
  - Tags_UpperLower (27.37%)
  - Tags_Lowercase (27.04%)
  - Tags_LowerUpper (26.91%)
  - WordNgrams_Existing_Unused (26.83%)
  - Tags_Uppercase (26.56%)
  - CharNgrams_Existing_Unused (26.16%)
  - WordNgrams_Existing_Numbers (26.01%)
  - WordNgrams_Existing_UpperNumber (25.90%)
  - WordNgrams_Existing_Lowercase (25.64%)
  - WordNgrams_Existing_Caps (25.39%)
  - Tags_UpperNumbers (25.33%)
  - CharNgrams_Existing_Numbers (24.64%)
  - Tags_LowerNumbers (24.59%)
  - CharNgrams_Existing_UpperNumber (22.81%)
  - CharNgrams_Existing_Caps (22.70%)
  - CharNgrams_Existing_Lowercase (22.66%)
  - CharNgrams_Existing_LowerNumber (22.65%)
  - Caps_UnusedPrefixes (2.45%)

5.7 XML vs JSON Empirical Results

This section compares the performance of XML and JSON data formats in the experiments overall, with implementation and results analysis the same as in Sections 5.5 and 5.6, respectively.

The results in Figure 7 and Table 6 demonstrate that XML was better than JSON for compression ratios and JSON was better than XML for compression and decompression times. These results were expected since verbose XML data would yield better compression ratios than the less verbose JSON data. Also, the compact nature of JSON data would produce faster compression and decompression processing times than larger XML data. Note that the results for the XMill experiments were excluded from this comparison in order to allow for a direct comparison between XML and JSON data formats.

Figure 7. Overall XML vs JSON Compression Results

Table 6. Average Compression Results for XML vs JSON

Data Format	Compression Ratio	Compression Time (s)	Decompression Time (s)
XML	0.171	0.898	0.167
JSON	0.191	0.609	0.117

5.8 SMILES Case Study Empirical Results

The following sections demonstrate the results for the SMILES Case Study experiments with implementation and results analysis the same as in Sections 5.5 and 5.6. Refer to Figures 1 to 12 and Tables 1 to 9 in Appendix E for the SMILES Case Study results highlighted in this section.

SMILES Comparison Compression Ratios

The following shows the data transforms that provided better overall compression ratios compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - Periodic Number (35.48%)
  - Caps (26.58%)
- WRT-LZ77
  - Periodic Number (35.53%)
  - Caps (26.64%)
- WRT-PAQ
  - Periodic Number (35.48%)
  - Caps (26.58%)
- WRT-PPM
  - Periodic Number (35.48%)

  - Caps (26.58%)
- Untransformed
  - Periodic Number (34.13%)
  - Caps (25.03%)

The following shows the data transform variations that provided better overall compression ratios compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - Reused NumPrefix PeriodicNum (42.08%)
  - Stars NumPrefix PeriodicNum (37.37%)
  - Reused Prefixes (32.28%)
  - Unused NumPrefix PeriodicNum (26.99%)
  - Unused Prefixes (20.87%)
- WRT-LZ77
  - Reused NumPrefix PeriodicNum (42.13%)
  - Stars NumPrefix PeriodicNum (37.42%)
  - Reused Prefixes (32.34%)
  - Unused NumPrefix PeriodicNum (27.05%)
  - Unused Prefixes (20.94%)
- WRT-PAQ
  - Reused NumPrefix PeriodicNum (42.08%)
  - Stars NumPrefix PeriodicNum (37.37%)
  - Reused Prefixes (32.28%)
  - Unused NumPrefix PeriodicNum (26.99%)
  - Unused Prefixes (20.87%)
- WRT-PPM
  - Reused NumPrefix PeriodicNum (42.08%)
  - Stars NumPrefix PeriodicNum (37.37%)
  - Reused Prefixes (32.28%)

  - Unused NumPrefix PeriodicNum (26.99%)
  - Unused Prefixes (20.87%)
- Untransformed
  - Reused NumPrefix PeriodicNum (40.86%)
  - Stars NumPrefix PeriodicNum (36.05%)
  - Reused Prefixes (30.86%)
  - Unused NumPrefix PeriodicNum (25.46%)
  - Unused Prefixes (19.21%)

SMILES Comparison Compression Times

The following shows the data transforms that provided better overall compression times compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - CharNgrams (0.34%)
- WRT-LZ77
  - CharNgrams (0.03%)
- WRT-PPM
  - CharNgrams (1.37%)
- Untransformed
  - CharNgrams (1.40%)

The following shows the data transform variations that provided better overall compression times compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - Unused (9.96%)
  - Existing_Unused (3.38%)
  - Existing_Numbers (1.12%)
- WRT-LZ77
  - Unused (9.68%)
  - Existing_Unused (3.08%)
  - Existing_Numbers (0.81%)
- WRT-PAQ
  - Unused (9.63%)
  - Existing_Unused (3.02%)
  - Existing_Numbers (0.75%)
- WRT-PPM
  - Unused (10.89%)
  - Existing_Unused (4.38%)
  - Existing_Numbers (2.14%)
- Untransformed
  - Unused (10.91%)
  - Existing_Unused (4.41%)
  - Existing_Numbers (2.16%)

SMILES Comparison Decompression Times

The following shows the data transforms that provided better overall decompression times compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - CharNgrams (1.51%)
- WRT-LZ77
  - CharNgrams (3.06%)
- WRT-PAQ

  - CharNgrams (0.71%)
- WRT-PPM
  - CharNgrams (1.16%)
- Untransformed
  - CharNgrams (2.18%)

The following shows the data transform variations that provided better overall decompression times compared to WRT and untransformed data. The percentage of how much they were better are shown in brackets:

- WRT-BWT
  - Unused (6.15%)
  - Existing_Unused (3.12%)
  - Existing_Numbers (1.86%)
  - Existing_LowerNumber (0.18%)
  - Existing_UpperNumber (0.01%)
- WRT-LZ77
  - Unused (7.63%)
  - Existing_Unused (4.64%)
  - Existing_Numbers (3.40%)
  - Existing_LowerNumber (1.75%)
  - Existing_UpperNumber (1.58%)
  - Existing_Caps (1.39%)
  - Existing_Lowercase (1.03%)
- WRT-PAQ
  - Unused (5.39%)
  - Existing_Unused (2.33%)
  - Existing_Numbers (1.06%)
- WRT-PPM
  - Unused (5.82%)
  - Existing_Unused (2.77%)

  - Existing_Numbers (1.51%)
- Untransformed
  - Unused (6.79%)
  - Existing_Unused (3.78%)
  - Existing_Numbers (2.53%)
  - Existing_LowerNumber (0.86%)
  - Existing_UpperNumber (0.69%)
  - Existing_Caps (0.50%)
  - Existing_Lowercase (0.13%)

5.9 Results Analysis Discussion

This section provides a discussion of the results obtained for the XML, JSON and SMILES case studies.

The results clearly indicate that using transforms over XML, JSON and SMILES data-specific data does improve compression. However, they also show that some transforms are better suited to providing better compression ratios, some are better to providing better compression times and others for decompression times. So, the data transforms can be selected according to your requirements in terms of the need for better compressed output size or processing times. In terms of data-specific transforms, the SMILES case study showed that the Periodic Number data-specific transform was better for compression rather than compression and decompression times. Refer to Figures 1 to 15 and Tables 1 to 24 in Appendix D for the XML and JSON results, and Figures 1 to 12 and Tables 1 to 9 in Appendix E for the SMILES Case Study results discussed sections 5.9.1 to 5.9.3.

5.9.1 Data Transforms

In terms of providing better compressed output size, the results showed that for both XML and JSON data formats, capital conversion was the best data transform, triumphing over the n-gram and tag substitutions. In particular, the results for XML showed that capital conversion was 16.38% better than character n-gram substitution, 23.95% better than tag conversions and 28.24% better than word n-gram substitution. The results for JSON showed that capital conversion was 19.52% better than tag conversions, 17.30% better than character n-gram and 26.64% better than word n-gram substitutions. This was due to this type of transform expanding the original data with the added prefixes, and the substitution of uppercase characters with existing lowercase characters. Character n-gram substitution was shown to provide better compressed output sizes compared to tag conversions by 9.05% for XML and 2.69% for JSON. This was expected since in these controlled experiments, the character n-gram substitutions were limited to a set of 26 n-gram substitutions per file, based on the most frequent n-grams, whereas, in the tag conversions, the dictionaries contained all the distinct element names for the substitutions which varied for each data file. Therefore, character n-gram transforms were better than tag conversions since they captured n-grams of the actual content as well as parts of the repetitive and verbose structure.

Tag conversions were better than the word n-gram substitutions by 5.64% for XML and 8.84% for JSON, since once again, as explained above, the distinct elements collected were based on all elements in the file structure and the number collected varied, compared to the fixed number of word n-grams collected, as per the character n-grams, of 26. In an ideal research set up, to allow for further extensive n-gram research to be conducted on such data, it would be far more beneficial and insightful to tailor and gather numbers of character and word n-grams, to the type of data being examined and the size and complexity of such data.

The benefits of capital conversion, character and word n-gram substitutions and tag conversion transforms for compressed output size have been discussed in

Section 2.7. The results for compression times showed that for both XML and JSON, both character and word n-gram transforms were faster at compressing files. The results for XML showed that tag conversions were 29.34% faster than capital conversions, and character n-gram and word n-gram substitutions were 33.99% and 51.45% better than capital conversions respectively. Character n-gram and word n-gram subsitututions were also 6.58% and 31.29% better than tag conversions respectively. Word n-gram substitutions were 26.45% better than character n-gram substitutions. The results for JSON showed that tag conversions were 29.27% better than capital conversions, along with character n-gram and word n-gram substitutions showing better results of 38.11% and 36.59% respectively. Character n-gram and word n-gram substitutions were also shown to have better results than tag conversions by 12.51% and 10.36% respectively. Character n-gram substitutions provided slightly better results than word n-gram substitutions by 2.39%. This was mainly due to the reduction in file sizes after transformation. On the other hand, the tag and capital conversions were slower in contrast due to larger file sizes following transformation. The decompression results for XML showed that word n-gram substitutions were 13.70% better than tag conversions; tag conversions were 31.06% better than capital conversions and character n-gram and word n-gram substitutions were 26.81% and 40.50% better respectively than capital conversions; tag conversions were 5.80% better than character n-gram substitutions and word n-gram substitutions were 18.70% better than char n-gram substitutions. The results for JSON showed that word n-gram substitutions were 0.77% better than tag conversions; tag conversions were 27.44% better than capital conversions and character and word n-gram substitutions were 25.77% and 28.00% better than capital conversions; and tag conversions were 2.25% better than character n-gram subsitutions compared to word n-gram substitutions which were 3.00% better. The ranking of the transforms in terms of decompression times were the complete reverse of compression ratios, signifying that compressed output size and decompression times are directly affected by each other. This information is useful to developers to ascertain how much querying times are impacted during decompression, which files would be

queried the most, which files should be transformed with which transform and transform variation, whether compressed output size or processing is more of a concern or both, and so on.

In applying data-specific transforms to SMILES data in the SMILES case study, the results demonstrated that the atomic element substitution faired better in compressed output size post compression, compared to the other general-purpose data transform techniques as applied to the XML and JSON datasets in previous experiments, namely, the capital conversion and character n-gram transforms. The results showed that the periodic number transform was 12.13% better than capital conversion; and that capital conversion and the periodic number transform were both 30.04% and 38.52% better respectively than the character n-gram substitution. The results for the data-specific transform were expected for SMILES data since the conversion of atomic elements to their corresponding numbers would reduce the number of distinct characters to compress. This result supports previous studies where data format-specific compressors have been developed, such as those discussed in Section 2.4 geared towards compressing XML data for instance, to enable compression of such data based on the structure of the data.

Although the transforms used in this thesis do include structure related transforms, such as tag conversions, other general purpose transforms have been used that focus on the data as a whole rather than just the structure. This is in contrast to data format-specific compressors, that focus mainly on the structure of the documents that they are designed to compress. The results from the experiments demonstrate that applying generic transforms to XML, JSON and data-specific SMILES data also improves compression. XML-specific compressors, on the other hand, only allow for compression of XML data, whereas, this thesis was concerned with developing both general-purpose transforms that could be applied to any type of data and data format, and also with developing domain-specific transforms using similar substitution and dictionary approaches, that allow transforms to be easily adopted and adaptable to domain-specific data.

As already mentioned, the results demonstrated that capital conversions was the next best result in terms of data compressed output size after

compression. Again, the number of distinct characters would be reduced and this transform, although a general-purpose transform, was particularly useful for SMILES data since atomic elements start with capital letters. The character n-gram transform provided the worst compressed output size overall, although character n-gram level 2, as expected for this type of data, provided very good compressed output size compared to the other n-gram levels, as will also be seen later. Coinciding with the trend for XML and JSON data formats, character n-gram substitutions also provided faster compression and decompression times for data-specific SMILES data. Leaving the atomic element substitution and capital conversion transforms behind in processing times. Specifically, the results showed that character n-gram substitution was faster than capital conversion during compression by 34.52%; and capital conversion was 14.90% faster than the periodic number transform along with character n-gram substitution by 44.28%. For decompression times the results demonstrated that character n-gram substitution was 15.04% faster than capital conversion; and both capital conversion and character n-gram substitution were 11.06% and 24.43% better than the periodic number transform.

In general, capital conversions and character n-gram substitutions further increased the transformed file sizes after transformation, compared to the tag and word n-gram substitutions, which decreased the transformed file sizes. This means that the compression ratios were improved during compression for the capital and character n-gram transformed files, compared to the tag and word n-gram transformed files. Also, for domain-specific data, such as SMILES, data-specific transforms, such as the atomic element substitution, can notably improve compression of the type of data the transform is tailored specifically for. Compression and decompression times were worse when good compressed output size was provided, and vice-versa, better processing times with worse compressed output size. For general and data-specific data, it can be noted that compressed output size and processing (compression and decompression) times impact each other. Better compressed output size compensates on processing times, better processing times compensate on compressed output size.

One could also argue that the conversion of XML data to JSON can be considered as a useful additional transform step in the compression process, since JSON was designed to provide a compact alternative to XML (refer to Section 2.5). The information contained in the results regarding JSON data will be beneficial for developers when looking at processing certain types of data. This extra step can establish itself as being useful when used in the whole selection process criteria.

The studies have shown that transforming files prior to compression can further extend file sizes, depending on which transform or transform variation (as will be discussed later) is being used. However, the results also establish that favourable compression results can nonetheless be successfully achieved. It is important to note that the process of transforming files can add extra processing times both prior to compression and after decompression. Although particularly for domain-specific data transforms, such as the atomic element substitution carried out for SMILES data, as long as the transforming dictionary is available to view, or in this case even if the dictionary was not available but the transforming technique and grammar was known, then it would be possible to use the transformed files in their transformed state (without the process of detransformation taking place), for querying purposes for example. This would be simple in this case since the information relating to atomic elements and their corresponding atomic numbers are widely available and easily located. A simple mapping is all it would take to process a file in their transformed state for SMILES data. This could be applied to other domain-specific data depending on the data-specific transforms being used and the method used to implement them.

For a simple dictionary approach, mapping information is all that is required to easily obtain the required information. The same could be applied to other general purpose transformed files from other data formats, such as XML and JSON. For example, in the tag conversions, particularly for querying purposes, if the tag information is known alongside its substitution characters, then the file could be subsequently parsed and queried in its transformed state.

The experiments have shown that relevant general purpose transforms can be applied to varied types of data from different data formats of varying sizes.

Some transforms can also be applied to domain-specific data to improve compression alongside data-specific transforms tailored to that type of data, such as the results from the SMILES case study.

5.9.2 Data Transform Variations

Further examination of the data transforms can be done on a granular level, with respect to the following:

- Existing and unused symbols
- Alphabetical, alphanumeric and numeric characters
- Uppercase and lowercase letters

Existing and Unused Symbols

In terms of providing better compressed output size after compression, the compression ratios for XML, JSON and SMILES data demonstrated that using existing characters for the addition of prefixes in the capital conversion transforms provided better compressed output size in comparison to their unused counterparts. Specifically, the results for XML, JSON and SMILES demonstrated that using existing prefixes for this transform were 4.43%, 5.01% and 14.42% better, respectively. The same conclusions from the results were encountered for all three sets of experiments for the character n-gram substitution, where the use of existing characters in the transformation process were better on the whole compared to unused characters. For this transform using existing characters was found to be 4% to 9% better than using unused characters for XML and JSON, and 5% to 10% for SMILES. This trend continued for the word n-gram substitution for JSON data showing an improvement of 2% to 5%. However, a slight deviation in this result was found in the word n-gram substitution, whereby for XML data on the whole, the results did show that the use of existing characters was predominantly better compared to unused characters with the results showing an improvement of 2% to 6%. However, the transform variation, Existing_Numbers, was found to be

slightly worse than the unused character variation by 1.08%. In the SMILES case study, the atomic element substitution transform demonstrated that reusing existing characters also benefited compressed output size for this type of data by 20.67% compared to unused characters. The star encoding transform variation was also shown to be better than unused characters by 14.21%.

The results for XML, JSON and SMILES demonstrated that reusing existing characters in the capital conversion transform provided slower compression times compared to unused characters, by 7.45%, 12.08% and 24.48% respectively. Similarly, the results for the character n-gram transform variations showed that reusing existing characters was 6% to 11%, 5% to 10%, and 7% to 15% slower respectively. The word n-gram transform variations showed that using existing characters provided up to 12% slower compression times for XML, and up to 2% slower results for JSON with the exception of the Existing_Lowernumber transform variation, which was 7.78% faster than the unused character transform variation. Also compared to the unused characters transform variation, the results for SMILES showed that using the existing character and the star encoding transform variations were 34.60% and 21.43% slower.

Decompression times provided similar results whereby all studies demonstrated that reusing existing characters was worse than the unused character transform variations. For XML, JSON and SMILES data, the capital conversion transform demonstrated that reusing existing characters was 8.46%, 8.25% and 14.45% worse than the unused character variations, respectively. The results for the character n-gram transform showed that the existing character variations were 6% to 12% slower for both XML and JSON, and 3% to 7% slower at decompression for SMILES. The results for the word n-gram transform showed that using existing characters was 3% to 22% slower for XML and up to 5% slower for JSON. For SMILES data, the atomic element substitution results demonstrated that using the existing character and star encoded transform variations provided slower decompression times by 25.91% and 15.18% respectively, compared to the unused character transform variation.

Alphabetical, Alphanumeric and Numeric Characters

For the tag conversion, alphanumeric transform variations provided up to 3% better compression ratios than alphabetical variations for XML and 2% for JSON. For character n-gram substitutions, it was found that alphanumeric variations were slightly worse than alphabetical by up to 0.15% for XML and up to 1% for JSON. Similarly, compression ratios were 3% worse for the numeric variation compared to alphabetical for both XML and JSON. For the SMILES case study, the alphanumeric data transform variations were found to provide slightly better compression ratios than the alphabetic variations by up to 1%, however, the numeric variation showed 3% worse compression ratios compared to the alphabetic variations. Also, for the word n-gram substitutions, alphanumeric transform variations were up to 1% better and numeric transform variations were up to 6% worse than the alphabetic variations for XML. However, the results for JSON demonstrated mixed compression ratio results for the alphanumeric and alphabetic transform variations, and slightly worse results for the numeric transform variation by 1% compared to the alphabetic variations.

Compression times for the tag conversion on both XML and JSON, showed that the alphanumeric transform variations were slower than the alphabetic variations by 6% for XML and 2% for JSON. The results for character n-gram substitutions demonstrated mixed results for alphanumeric and alphabetic transform variations for XML, and a 1% improvement in compression times for alphanumeric variations for both JSON and SMILES. The numeric variations compared to alphabetic variations showed 2%, 3% and 4% improvements in compression times for XML, JSON and SMILES, respectively. The word n-gram substitution results for the alphanumeric variations showed that they were worse by 3% for XML and better by up to 10% for JSON, compared to alphabetic variations. The numeric variation for this transform showed that XML was 9% worse and JSON was up to 1% better than the alphabetic variations.

Decompression times for tag conversion demonstrated that the alphanumeric transform variations were 3% to 6% slower for XML and 1% to 4% slower for JSON, than the alphabetic variations. The results for the character n-

gram substitution demonstrated mixed results between alphanumeric and alphabetic variations for both XML and JSON. Whereas, the results for SMILES showed a 1% improvement to the alphanumeric variation. The results for the numeric transform variation showed 3%, 2% and 2% improvements compared to the alphabetic variations, for XML, JSON and SMILES respectively. The word n-gram substitution transform variations showed mixed results among the alphanumeric, alphabetic and numeric transform variations for XML. Whereas, JSON demonstrated slightly better results for the alphanumeric variations by up to 5% and for the numeric variations by 1%.

Uppercase and Lowercase Letters

The compression ratio results for the tag conversion transform demonstrated that the uppercase transform variations were slightly better than lowercase by up to 0.05% for XML, however, JSON provided mixed results. The improvements were also negligible for the character n-gram uppercase variations by up to 0.1% for both XML and JSON, however, SMILES provided mixed results for this transform. The results for the word n-gram substitution also did not form a pattern with its mixed results for both uppercase and lowercase variations for XML and JSON.

Compression times for the tag conversion transform showed only slight improvements in speed for XML by up to 0.2% for the lowercase transform variations compared to the uppercase variations, however, JSON provided mixed results. Whereas, the negligible improvements for the character n-gram lowercase variations were up to 0.2% for JSON and up to 0.5% for SMILES compared to the uppercase variations, however, XML provided mixed results for this transform. The results for the word n-gram substitution demonstrated that uppercase was slightly faster than lowercase for XML by up to 2% and no pattern was formed for JSON due to its mixed results.

Decompression times showed no pattern between uppercase and lowercase in the tag conversion transform for both XML and JSON. Both XML and SMILES also formed no pattern in the character n-gram substitutions, whereas,

improvements to the uppercase variations were up to 0.2% for JSON. The results for the word n-gram substitution demonstrated that lowercase variations were faster than uppercase variations for both XML and JSON by up to 15% and 4% respectively.

The principle and benefits of reusing characters for substitutions and also for added prefixes in the data transform variations to improve compression was taken from the star encoding scheme described in Section 2.7.2. This has shown that compression can be improved due to the reduction in the number of distinct characters to compress. Prefixes are also useful to ensure data integrity is kept intact and to avoid ambiguity on detransformation. These granular approaches to data transform variations have been shown in these experiments to improve compression for XML, JSON and domain-specific SMILES data.

5.9.3 Character and Word N-Gram Transform Levels

In terms of providing better data compressed output size, XML data provided better compression ratios for character n-gram levels 2 and 3 by 14.55% and 6.82% respectively, levels 4 to 7 were only marginally better by up to 3% and levels 8 and 9 were only up to 0.5% worse compared to level 10, whereas, for word n-gram substitutions, all levels were 14% to 19% better compared to word n-gram level 2. The results for JSON data showed that compression ratios were better for character n-gram levels 2, 3 and 4 by 15.66%, 7.92% and 5.53% respectively, and levels 5 to 9 were up to 3% better compared to level 10, whereas, for word n-gram all levels provided a 2% to 5% improvement in the compressed output size compared to level 1. For SMILES data, character n-gram level 2 was better by 8.38% and the rest of the levels were up to 7% worse than level 10. Better compression ratios for character n-gram level 2 overall was expected due to the higher frequencies of these particular n-grams compared to the other character n-gram levels. In contrast, better compression ratios were generally provided for the higher word n-gram levels for both XML and JSON, this was due to these n-grams consisting of words relating to the XML and JSON document structures. Since XML

is more verbose than JSON, XML showed greater improvements in compression ratios for the higher word n-gram levels generally compared to JSON.

Compression times for XML show that character n-gram levels 3 to 8 are 9% to 13% better than level 2, and that levels 9 and 10 are 28% to 29% better respectively, whereas, JSON shows that all levels are 10-13% better than level 2 and SMILES demonstrates that all levels are 12% to 19% better than level 2. For the word n-gram substitution, n-gram level 2 was 17.22% better and the other levels were 2% to 3% better than level 3 for XML. For JSON, word n-gram levels 1, 2, 3, and 5 were 9.96%, 4.18%, 6.46% and 1.28% better than level 4, respectively. Decompression times were shown to be similar to compression times for XML, whereby character n-gram levels 3 to 8 for XML were 9% to 14% better than level 2 and levels 9 and 10 were 24% better; the same similarity can be found in the results for JSON, whereby all levels were 8% to 15% better than level 2. The XML decompression results for word n-gram levels demonstrated that level 2 was 16.98% better than level 5, and the other levels were only marginally up to 1.6% better than level 5. The results for JSON showed that levels 1, 2, 3 and 5 were 7.53%, 3.52%, 5.16% and 1.73% better than level 4, respectively. The decompression results for SMILES showed that levels 3 to 6 were faster by 7% to 10% in ascending order compared to level 2 and levels 7 to 10 were 8% to 9% faster than level 2 in descending order.

Both character and word n-gram level experiments showed that after a certain point there was not much improvement to compression, where it levels off in the graphs. This finding is supported in literature [61] in reference to n-grams. Although the n-gram research conducted in this study contributes to existing research discourse in this area, as it shows how both character and word n-gram levels compare when using compression over transformed files using various transforms, compared to the previous n-gram literature which generally shows n-grams being tested on untransformed data. This could be a fruitful area for future work.

5.9.4 Balancing Compressed Output Size and Processing Times

In one form or another, the results did demonstrate some balance between compressed output size and processing times, whether this was between compressed output size and compression times or decompression times, or between compression and decompression times. These are based on the observed overall graphs for compression algorithms as shown in Appendices D and E.

The results for XML demonstrated that a balance existed for the 7Zip compression algorithm between compression ratios and decompression times as they were 93.79% and 96.77% better than compression times, respectively. Similarly, PPMd also illustrated a balance between these two metrics as they were 26.68% and 28.33% worse than compression times, respectively. PPMVC showed a balance between compression ratios and compression times as they were 45.28% and 42.07% better than decompression times. XMill-PPMdi demonstrated a balance between compression times and decompression times since they were both 415% worse than compression ratios. Looking at cases where the results showed some balance, which was defined as up to 20% difference between the metrics, BZip2 showed some balance between compression ratios and decompression times since they were 90.92% and 76.45% better than compression times respectively. XMill-BZip2 demonstrated some balance between these two metrics as they were 87.88% and 76.31% better than compression times respectively. Similarly, XMill-GZip also demonstrated some balance in these metrics as they were 71.94% and 83.23% better than compression times.

The results for JSON showed a balance for 7Zip and BZip2 between the compression ratios and decompression times metrics. The results showed that for 7Zip these metrics were 89.81% and 96.60% better than compression times respectively, and for BZip2 they were 83.07% and 76.66% better than compression times respectively. PPMVC demonstrated a balance between compression ratios and compression times since they were 27.38% and 32.64% better than decompression times respectively. PPMd showed some balance between

compression times and decompression times as they were 42.35% and 28.96% better than compression ratios respectively.

SMILES showed a balance between the compression and decompression times metrics for the PPMd compression algorithm. Compression times were 69.70% and decompression times were 63.83% better than compression ratios.

Further analysis of the compression metrics demonstrated that, overall for XML, JSON and SMILES data, negative correlations existed for both compression and decompression times when compared against compression ratios for all data transforms, data transform variations, character n-gram and word n-gram levels. However, no correlation could be deduced from the compression algorithms results. The analysis also showed that positive correlations existed between decompression and compression times when compared against each other for all data and scenarios, except for compression algorithms. Refer to Figures 16 to 25 and Tables 25 to 29 in Appendix D for the XML results; Figures 26 to 35 and Tables 30 to 34 in Appendix D for the JSON results; and Figures 13 to 20 and Tables 10 to 13 in Appendix E for the SMILES Case Study results discussed here.

These insights are useful since it can be seen that providing better compressed output size can have an adverse effect on either compression or decompression or both of these processing times, and vice-versa. This information is key to providing developers with guidance on the best compression algorithm to use for their data, depending on whether they require better compression ratios, compression times or decompression times, or a combination of these. An ideal situation would be to be able to balance both compressed output size with processing times where possible.

5.9.5 Statistical Significance Testing

The null (H_0) and alternative (H_A) hypotheses stated below were to test the effect of compression algorithms, data transforms and data formats on the compression metrics. Multivariate Analysis of Variances (MANOVAs) were selected over conducting a series of multiple Analysis of Variances (ANOVAs) as the most suitable statistical tests for all hypotheses to reduce the probability of making a Type I error, and also to see the effect of the results on the combination of the compression metrics [29]. The statistical tests were carried out using SPSS [74] and the results were as follows:

XML

- (H_0): The XML results of the combination of average compression ratios, average compression times and average decompression times metrics do not differ by compression algorithms and data transforms.

- (H_A): The XML results of the combination of average compression ratios, average compression times and average decompression times metrics do differ by compression algorithms and data transforms.

To investigate differences among the nine compression algorithms, comprising of six general-purpose and three XMill compression algorithms, and four data transforms, the following three dependent variables were entered into a multiple-factor between-subjects MANOVA: average compression ratios, average compression times and average decompression times. Non-significant multivariate effects were found for compression algorithms, data transforms and the interaction between compression algorithms and data transforms. Therefore, the null hypothesis was not rejected and it was assumed that the scores on the combination of the compression metrics did not differ by compression algorithms and data transforms.

JSON

- (H_0): The JSON results of the combination of average compression ratios, average compression times and average decompression times metrics do not differ by compression algorithms and data transforms.

- (H_A): The JSON results of the combination of average compression ratios, average compression times and average decompression times metrics do differ by compression algorithms and data transforms.

To investigate differences among the six general-purpose compression algorithms and four data transforms, the following three dependent variables were entered into a multiple-factor between-subjects MANOVA: average compression ratios, average compression times and average decompression times. Non-significant multivariate effects were found for compression algorithms, data transforms and the interaction between compression algorithms and data transforms. Therefore, the null hypothesis was not rejected and it was assumed that the scores on the combination of the compression metrics did not differ by compression algorithms and data transforms.

SMILES Case Study

- (H_0): The SMILES Case Study results of the combination of average compression ratios, average compression times and average decompression times metrics do not differ by compression algorithms and data transforms.

- (H_A): The SMILES Case Study results of the combination of average compression ratios, average compression times and average decompression times metrics do differ by compression algorithms and data transforms.

To investigate differences among the six general-purpose compression algorithms and three data transforms, comprising of two general-purpose data transforms and one data-specific transform, the following three dependent variables were entered into a multiple-factor between-subjects MANOVA: average compression ratios, average compression times and average decompression times. Non-significant multivariate effects were found for compression algorithms, data transforms and the interaction between compression algorithms and data transforms. Therefore, the null hypothesis was not rejected and it was assumed that the scores on the combination of the compression metrics did not differ by compression algorithms and data transforms.

XML vs JSON

- (H_0): The XML and JSON results of the combination of average compression ratios, average compression times and average decompression times metrics do not differ by compression algorithms, data transforms and data formats.

- (H_A): The XML and JSON results of the combination of average compression ratios, average compression times and average decompression times metrics do differ by compression algorithms, data transforms and data formats.

In order to provide a direct comparison between XML and JSON data formats the XMill experimental results were excluded from this MANOVA test. To investigate differences among the six general-purpose compression algorithms, four data transforms and two data formats, the following three dependent variables were entered into a multiple-factor between-subjects MANOVA: average compression ratios, average compression times and average decompression times. Non-significant multivariate effects were found for compression algorithms, data transforms, data formats, the interaction between compression algorithms and data transforms, compression algorithms and data formats, data transforms and

data formats, and between the interaction between data algorithms, data transforms and data formats. Therefore, the null hypothesis was not rejected and it was assumed that the scores on the combination of the compression metrics did not differ by compression algorithms, data transforms and data formats.

5.10 Chapter Summary

This chapter provided detailed results from a comparative analysis of XML and JSON data using tag document structure, capital letter, character and word-n-gram transforms, with a variety of transform variations that included:

- Alphabetical, alphanumeric and numeric characters
- Uppercase and lowercase letters
- Existing and unused symbols

For the SMILES case study, the relevant capital letter and character-n-gram transforms were adopted and a domain-specific transform was used. In this case, this was the atomic number substitution. As described previously, transform variations were also used in this case study. All results for the XML study were compared with XMill and the results for all studies were compared with the WRT transforms and untransformed data.

This chapter also discussed these results further in terms of the best compression compressed output size and processing time results for data transforms, data transform variations, character and word n-gram levels, balanced compressed output size and processing times and statistical significance.

All studies demonstrated that using some data transforms and transform variations over XML, JSON and SMILES data did improve compression ratios, compression and decompression times when compared to XMill, WRT and untransformed data. The studies also showed that using the different transform variations did provide some extra compression benefits. XML generally provided

better compression ratios than JSON, and JSON provided better compression and decompression times compared to XML. An analysis of the different compression metrics with compression algorithms identified which compression algorithms were balanced according to their compression ratios, compression and decompression times. The compression algorithms 7Zip, BZip2, PPMd and PPMVC, in particular, were identified as balanced for the XML and JSON studies. Further analysis of the compression metrics highlighted a correlation between compression times and decompression times for all studies. Finally, the results for the MANOVA statistical tests carried out for all hypotheses stated, demonstrated that these results were not statistically significant and thus all the null hypotheses were accepted. The next chapter concludes this thesis.

Chapter 6

Conclusions and Future Work

6.1 Introduction

The previous chapter presented, analysed and discussed the results of both the XML and JSON main study and the SMILES case study.

This chapter concludes this thesis with the following:

- Research outcomes
- Practical and theoretical contributions
- Study limitations and future work

6.2 Thesis Contributions

Research Outcomes

This study demonstrated, with transforms and a number of different transform variations, that using general-purpose compressors over transformed data can effectively improve compression further, whether that be improvement of data compressed output size or processing times or both. The results also revealed that in some cases, the transforms and transform variations developed in this study were better than using an existing XML-Specific compressor, such as XMill, and an existing general-purpose transform technique, namely WRT, and also

better than the results for compression over untransformed data. Similar improvements were noted for the results of the SMILES case study, using both data-specific and relevant general-purpose transforms. In general, it can be concluded that transforms developed for both generic and specific data can enhance compression when used with other compressors.

The experimental approach in this study shows that general purpose compressors and transforms and transform variations can be applied to any type of data of any format and tailored accordingly, generally without restrictions, compared to other existing techniques, such as XML-specific compressors, and so on. The results from both XML and JSON formats can be generalised across results from similar data formats, since XML and JSON formats are fairly similar in terms of the need for a structured document with element tags. However, the structure of a JSON document is far less verbose in comparison to an equivalent XML document, therefore, this suggests that the results from these data formats can be generalised with similar data formats to XML and JSON, in terms of document structure. It could also be argued that due to the less verbose nature of JSON documents, which results in JSON files being more text based compared to XML documents, the results from the JSON set of experiments could also be somewhat generalised, with some caution in areas, with data that contains less structure or even with no structure; just textual data. Data from the SMILES case study, on the other hand can closely be generalised with other SMILES data available in databases within the Chemoinformatics industry.

This has wide implications in industry today where data is continuously growing, where data types and data formats are being developed and extended. It is often the case that developers and researchers at times would benefit from different approaches to compression, such as the techniques used in this research, to be able to make key decisions about compressed output size and processing of any type of data.

Practical Contributions

It was rapidly uncovered from this study that it was not going to suggest a one-size-fits-all solution, unless it narrowed its focus solely on specific compressed output size or processing requirements, compression algorithms, data groups, transforms, transform variations, n-gram levels, and even more on a granular level, such as alphanumeric transform variations, and so on. Since this study focused on a number of factors, the results were able to provide richer insights that can be used to assist developers and researchers in making the following decisions, depending on their compression requirements, such as better compressed output size costs and compression processing times and query (decompression) processing times; using the results from the compression ratio, compression and decompression time metrics, respectively, or a balance in compressed output size and processing costs:

- The best, both overall and on a lower level, data formats, compression algorithms, data transforms, data transform variations, character and word n-gram levels, for the type of data (identified by data groups in the experiments) that needs to be compressed.
- Whether it would be beneficial to include the conversion of XML to JSON as a potential additional part of the process for some types of data.
- The best, both overall and on a lower level, domain-specific compression algorithms, data transforms, data transform variations, character n-gram levels, for the type of data that needs to be compressed. In this case, for SMILES data.
- Whether it would be useful to use other existing XML-Specific compressors, such as XMill used in the XML study, or other existing data transform techniques, such as WRT used in all studies, or even in some cases leave the data in its untransformed state, in place of the transforms and transform variations developed in this thesis, for some types of data.
- The best computing resources necessary in order to achieve the required compression goals.

The experiments were conducted using a variety of different types of data groups across many classes, ranging from auction or bidding data, to toxicology data, to a large set of wikimedia datasets, as well as domain-specific SMILES data. The use of this diverse set of datasets implies that the results for this study can benefit both developers and researchers across a wide range of disciplines, in adopting and tailoring these techniques specifically for the data that they are using. Developers and researchers in the Chemoinformatics discipline can certainly benefit from the results in the SMILES case study. As mentioned above, results from the XML and JSON set of experiments can also provide further guidance on whether or not it would be beneficial to convert XML to JSON, since there are a number of conversion tools that enable this process, in order to achieve optimum compression. The results from all experiments carried out in these studies also provide guidance for developers and researchers if data should be used with other existing XML-Specific compressors, such as XMill, or other existing transform techniques, such WRT, or even remain in their untransformed state to achieve their desired compression. However, the results did confirm that compression over some of the transforms and transform variations developed in this thesis, were actually better than compression over than WRT and untransformed data, in a number of cases for XML, JSON and also in the SMILES case study.

The essence of this study was to contribute complete, unbiased information and results to enable developers and other researchers in this area to compress data appropriately according to their needs. To this end, the results have portrayed that different transforms will work better with different transform variations (including character and word n-gram levels), across different compression algorithms, data groups and data formats. The same has been seen in the SMILES case study. These results are beneficial for different compression requirements, in terms of providing better compression ratios, compression times and decompression times (refer to the results in Sections 5.5 to 5.8).

Theoretical Contributions

The inclusion of data transforms and variations could be extended to other existing transform techniques, for example XML-specific techniques could be further improved. In an attempt to reduce the structural redundancy of XML data, some existing XML-specific techniques used schemas, such as an XML Schema, (refer to Sections 2.4.2 and 2.4.4 in the literature review). Since the general-purpose data transforms and transform variations used in both the main study and case study are flexible and can be easily adapted, these transform techniques could be integrated with a schema to potentially improve the compression of XML data. In order to cater for JSON data, the schemas would need to be adaptable to work with other data formats. However, the suggested improvement implies working with a text based schema, which incurs compressed output size costs, compared for example to a binary based schema developed in [28]. The information present in existing schemas that XML documents conform to, could also be used as guidance for researchers and developers to determine what type of data transform to use and the best type of data transform variation to use to provide efficient compression of XML data. This information could be used alongside the results from this study for those documents that conform to schemas, however, not all XML documents have a schema attached to them [71], [70], [8].

6.3 Study Limitations and Future Work

In addition to some of the suggestions to improve the study discussed in the Methodology Chapter 3, study limitations and future work is discussed in the following areas.

Data Exchange Formats

This study could be further extended and compared to other data exchange formats, such as YAML and BSON, as mentioned in the literature review. The results from these extended experiments would appeal to developers using these

data exchange formats.

Variations in Datasets

Another improvement to this study would be to investigate the nature of the data being compressed in order to provide additional insight into what type of data affects the efficacy of the various compression and data transform algorithms. This would lead to the development of a meta-algorithm that could be used to analyse the data in terms of its characteristics to help provide better compression of different types of data. A schema could also be used alongside this algorithm to provide the necessary information about an XML document.

Data Transform Techniques

The punctuation marks modelling technique was not used in the study due to the excessive run times of other data transforms used and was described in Section 2.7.5, however, it could have produced fruitful results if it were to be used to facilitate word prediction in the main study. However, whilst punctuation marks modelling itself is certainly more beneficial for textual documents that contain punctuation marks, such as the datasets in the Wikimedia and Shakespeare data groups for example, the concepts used in this transform could also be adapted to other types of data. An example is the SMILES case study. For this case study, instead of treating SMILES strings as one “word” in data transforms, the punctuation marks modelling technique could also have been adapted and tailored towards the vocabulary used in SMILES data to consider other symbols, such as double or triple bonds, branches, and so on. This could then have been combined with the space stuffing technique, to separate SMILES strings into segments and then produce word predictions for word n-gram data transforms of SMILES data.

The studies focused on results for individual transforms to give researchers and developers information on these transforms, and flexibility for them to make choices on which transforms and transform variations to select for their needs, and which transforms to combine for further compression benefits if they wish to. Combining data transforms and transform variations was not conducted since it

was considered beyond the scope of this study. However, this inclusion to this study would provide further compression benefits for future work, but with a cost to processing times. In progression of this study, future work could experiment with different data transform combinations, and provide researchers and developers with key knowledge of the best data transform combinations to use with different types of data. This could also have other benefits, such as also improving word predictions with word n-grams, by for instance, collecting n-grams after the tag transformation to ensure n-grams remained focused on the content of the document and not on the verbose document structure, or after another combined set of transforms that can be used to improve n-gram data transformation for further compression benefits.

A further improvement to the n-gram collection phase used in this study could be to include word stemming in the approach as described in [17], [61]. This essentially would allow for similar forms of words to be grouped together. For example, the words ‘compress’, ‘compressed’, ‘compressing’ and ‘compression’ have the words ‘compress’ in common. Whilst this process may incur extra processing costs in terms of both n-gram collection and data transform processing times, the knowledge could certainly be used to improve n-gram collections, and thus provide further improvement to compression through better n-gram data transforms. The n-gram section of this study could also be combined with research in n-gram similarity [17] as a future research direction, for the purposes of improving data compression.

Both character and word n-gram level research, related to data compression in itself, would be another interesting research direction, particularly in relation to the use of different types of data, and also working with similar groups of data.

In this study, word n-gram collections took a while to process, particularly on the larger datasets; this implies that a better processor is required when considering word n-gram data gathering. The larger files considered but not included due to processing errors, as mentioned in Section 5.2.1, could have been included in this study if a better processor with potentially more memory was available.

Compression Algorithms

A more comprehensive, but exhaustive study could include further compression algorithms to be used over the transformed data, for example, similar to the wide range of algorithms tested in [49]. However, this would require more time for experimental and analysis purposes, as well as better computing resources in terms of processors and memory capacity. The general-purpose compressors used in this study were selected from statistical, dictionary-based and transform techniques, are well-known and were within the scope of this study, default compressor settings were used, however. Future work could involve using other compressor settings to provide further comparisons and insights.

Metrics and Benchmarks

The data transformation results, transformed file sizes and data transformation processing times, were not included as part of this study, to focus the whole thesis on compression over transformed data. The analysis of transformation results combined with this study, would provide researchers and developers with more insights into the cost of the transformation and detransformation processes, prior to and post compression, as well as the compression results. However, the focus on compression in this thesis would enable researchers and developers to see which transforms and variations provide the compression they are seeking, for the type of data they are working with. Then they can select the transforms and variations and investigate further from there. Whereas, focusing on both transformation and compression results may affect their choices, particularly with the extra preprocessing and postprocessing times that some transforms and transform variations incur and compressed output size costs. However, the inclusion of transformation results could be included in future work to improve this study.

The reasons for only including one XML-specific compression technique, namely XMill, were mentioned in Section 3.2.11, which were due to the unavailability of many of these techniques and comparative limitations with other

data formats if they were to be used in this study. However, other general-purpose benchmarks could be used in the future, for example LIPT and StarNT that were described in Section 2.7.2., to further improve this study.

Case Studies

As mentioned previously, some XML-specific techniques group similar data together and place them in containers, such as XMill [5], [9], [14], [46], [7], [45] which group data together based on the structural information contained in XML documents. This is analogous to the data used in the SMILES case study, whereby data transforms and transform variations were applied to one specific group of data. Further case studies could therefore provide further insights into the extent that using data transforms and transform variations could improve the compression of similar types of data. With the variety of datasets used in this thesis, a number of different interesting case studies could be conducted on the foundation of these data transforms and transform variations.

Another, more practical, approach to a case study would be to collaborate this research with a developer in industry in order to further advance this study to a more focused point of view. This would entail a full requirements analysis, working with developers to find out their particular needs relating to the type of data they work with, their expectations relating to compressed output size and processing times, and any other requirements that they may have relating to their data. Research would be required into the appropriate methods to test and carry out a wide set of experiments on their data, to provide them with the best transforms, transform variations, compressors, other recommendations, and other resources and information required to fulfil their requirements. This focal point would also permit the use of combined efficiency compression metrics, such as those described in [68], [10], [66], [49], or even the development of a new combined efficiency metric that could be tailored to the needs specified by the developer.

The SMILES case study could be extended in the future to work with a larger amount of SMILES data, if the data was easily obtainable. There is also the potential to collaborate with researchers in this domain to experiment with the data

they use. Other extensions to this study could be to tailor and apply the transforms and transform variations to other chemical notations, for example those that possess lexical similarities with SMILES notations, such as SMARTS [23], which is an extension of SMILES data.

Queriability

Queriability is another area for future research, where data could be potentially queried in their transformed state to avoid incurring extra processing times with detransformation.

References

1. 7z Format. [Online]. Available: http://www.7-zip.org/7z.html [Accessed: 12 November 2012]
2. Abel, J., Teahan, W.: Universal Text Preprocessing for Data Compression. IEEE Transactions on Computers. 54(5), 497–507 (2005)
3. Al-Laham, M., El Emary, I.M.M.: Comparative Study Between Various Algorithms of Data Compression Techniques. International Journal of Computer Science and Network Security (IJCSNS). 7(4), 281-291 (2007)
4. Alkhatib, R., Scholl, M.H.: CXQU: A Compact XML Storage for Efficient Query and Update Processing. In: Proceedings of the 3rd International Conference on Digital Information Management (ICDIM 2008), London, England, November 13-16, pp. 605-612 (2008)
5. Alkhatib, R., Scholl, M.H.: Efficient Compression and Querying of XML Repositories. In: Proceedings of the 19th International Workshop on Database and Expert Systems Application (DEXA 2008), Turin, Italy, September 01-05, pp. 365-369 (2008)
6. Alom, B.M.M., Henskens, F, Hannaford, M.: Querying Semistructured Data with Compression in Distributed Environments. In: Proceedings of the 6th International Conference on Information Technology: New Generations (ITNG 2009), Las Vegas, Nevada, USA, April 27-29, pp. 1546-1553 (2009)
7. Arion, A., Bonifati, A., Costa, G., D'Aguanno, S., Manolescu, I., Pugliese, A.: Efficient Query Evaluation Over Compressed XML Data. In: Proceedings of Advances in Database Technology, the 9th International Conference on Extending Database Technology (EDBT 2004), Iraklion, Greece, March 14-18, Lecture Notes in Computer Science 2992, Springer, Berlin, pp. 200-218 (2004)

8. Arion, A., Bonifati, A., Manolescu, I., Pugliese, A.: Path Summaries and Path Partitioning in Modern XML Databases. World Wide Web – Internet and Web Information Systems. 11(1), 117-151 (2008)
9. Arion, A., Bonifati, A., Manolescu, I., Pugliese, A.: XQueC: A Query-Conscious Compressed XML Database. ACM Transactions on Internet Technology. 7(2), Article No. 10 (2007)
10. Augeri, C.J., Mullins, B.E., Baird III, L.C., Bulutoglu, D.A., Baldwin, R.O: An Analysis of XML Compression Efficiency. In: Proceedings of the ACM Workshop on Experimental Computer Science, San Diego, CA, USA, June 13–14, Article No. 7 (2007)
11. Bottcher, S., Hartel, R., Heinzemann, C.: BSBC: Towards a Succinct Data Format for XML Streams. In: Proceedings of the 4th International Conference on Web Information Systems and Technologies (WEBIST 2008), Funchal, Portugal, May 04-07, pp. 13-21 (2008)
12. Bottcher, S., Hartel, R., Messinger, C.: XML Stream Data Reduction by Shared KST Signatures. In: Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS 2009), Big Island, Hawaii, January 05-08, pp. 1-10 (2009)
13. Brisaboa, N.R., Cerdeira-Pena, A., Navarro, G.: XXS: Efficient XPath Evaluation on Compressed XML Documents. ACM Transactions on Information Systems (TOIS). 32(3), 13 (2014)
14. Buneman, P., Grohe, M., Koch, C.: Path Queries on Compressed XML. In: Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003), Berlin, Germany, September 09-12, pp. 141–152 (2003)
15. BZip2 for Windows. [Online]. Available: http://gnuwin32.sourceforge.net/packages/bzip2.htm [Accessed: 12 November 2012]
16. Carus, A., Mesut, A.: Fast Text Compression Using Multiple Static Dictionaries. Journal of Information Technology. 9(5), 1013–1021 (2010)

17. Cavnar, W.B., Trenkle, J.M.: N-Gram-Based Text Categorization. In: Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR 1994), Las Vegas, USA, pp. 161–175 (1994)
18. Chapin, B., Tate, S.R.: Higher Compression from the Burrows–Wheeler Transform by Modified Sorting. In: Proceedings of the IEEE Data Compression Conference (DCC 1998), Snowbird, Utah, USA, March 30-April 01, pp. 532 (1998)
19. Cheney, J.: Tradeoffs in XML Database Compression. In: Proceedings of the Data Compression Conference (DCC 2006), Snowbird, Utah, USA, March 28-30, pp. 392- 401 (2006)
20. Cherukuri, K., Agarwal, S.: XAdap: An Adaptive Huffman Coding on Markup Languages. In: Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, Sivakasi, Tamil Nadu, India, December 13-15, pp. 147-151 (4) (2007)
21. Compression.ru Project (in Russian). [Online]. Available: http://www.compression.ru/ds/ [Accessed: 12 November 2012]
22. Dawy, Z., Hagenauer, J., Hoffmann, A.: Implementing the Context Tree Weighting Method for Content Recognition. In: Proceedings of the IEEE Data Compression Conference (DCC 2004), Snowbird, Utah, USA, March 23-25, pp. 536 (2004)
23. Daylight Theory Manual. [Online]. Available: http://www.daylight.com/dayhtml/doc/theory/index.html [Accessed: 12 November 2012]
24. DSSTox. [Online]. Available: http://www.epa.gov/ncct/dsstox/ [Accessed: 15 October 2012]
25. Efficient XML Interchange Working Group. [Online]. Available: http://www.w3.org/XML/EXI/ [Accessed: 22 January 2012]
26. Engel, T.: Basic Overview of Chemoinformatics. Journal of Chemical Information and Modeling. 46(6), 2267–2277 (2006)
27. ExpoCast. [Online]. Available: http://www.epa.gov/ncct/expocast/ [Accessed: 15 October 2012]

28. Fang, J., Martinez-Smith, A., Gandhi, B.: A Compressed XML Schema Representation for Metadata Processing in Mobile Environments. In: Proceedings of the 8th IEEE Workshop on Multimedia Signal Processing, Victoria, Canada, October 03-06, pp. 493-496 (2006)
29. Field, A.: Discovering Statistics using SPSS (3rd ed.). London: Sage Publishing (2009)
30. File Splitter. [Online]. Available: http://sourceforge.net/projects/file-splitter/ [Accessed: 23 March 2014]
31. Girardot, M., Sundaresan, N.: Millau: An Encoding Format for Efficient Representation and Exchange of XML Over the Web. In: Proceedings of the 9th International World Wide Web Conference (WWW '00), Amsterdam, The Netherlands, May 15-19, pp. 747–765 (2000)
32. Gou, G., Chirkova, R.: Efficiently Querying Large XML Data Repositories: A Survey. IEEE Transactions on Knowledge and Data Engineering. 19(10), 1381-1403 (2007)
33. Hausenblas, M., Villazón-Terrazas, B., Cyganiak, R.: Data Shapes and Data Transformations. CoRR abs/1211.1565 (2012)
34. Homepage of Przemysław Skibiński. [Online]. Available: http://pskibinski.pl/ [Accessed: 12 November 2012]
35. Hruska, P., Martinovic, J., Dvorsky, J., Snasel, V.: XML Compression Improvements Based on the Clustering of Elements. In: Proceedings of the International Conference on Computer Information Systems and Industrial Management Applications (CISIM 2010), Krackow, Poland, October 08-10, pp. 217-221 (2010)
36. Index of /~anhai/wisc-si-archive/data/courses. [Online]. Available: http://pages.cs.wisc.edu/~anhai/wisc-si-archive/data/courses/ [Accessed: 22 January 2012]
37. Index of niagara/data/plays. [Online]. Available: http://research.cs.wisc.edu/niagara/data/plays/ [Accessed: 22 January 2012]
38. Introducing JSON. [Online]. Available: http://json.org/ [Accessed: 10 June 2015]

39. JSON and BSON. [Online]. Available: https://www.mongodb.com/json-and-bson [Accessed: 10 June 2015]
40. JSON: The Fat-Free Alternative to XML. [Online]. Available: http://www.json.org/xml [Accessed: 12 March 2014]
41. Karthikeyan, M., Bender, A.: Encoding and Decoding Graphical Chemical Structures as Two-Dimensional (PDF417) Barcodes. Journal of Chemical Information and Modeling. 45(3), 572–580 (2005)
42. Kristensen, T.G., Nielsen, J., Pedersen, C.N.S.: Using Inverted Indices for Accelerating LINGO Calculations. Journal of Chemical Information and Modeling. 51(3), 597–600 (2011)
43. Kurita, H., Hatano, K., Miyazaki, J., Uemura, S.: Efficient Query Processing for Large XML Data in Distributed Environments. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications (AINA 2007), Niagara Falls, Canada, May 21-23, pp. 317-322 (2007)
44. Li, W.: Xcomp: An XML Compression Tool. MSc Thesis, University of Waterloo, Ontario, Canada (2003)
45. Liefke, H., Suciu, D.: XMill: An Efficient Compressor for XML Data. In: Proceedings of the International Conference on Management of Data, Dallas, Texas, USA, May 15-18, pp. 153-164 29(2) (2000)
46. Lv, J., Wang, Y. Zhong, Y.: Efficient XML Document Compressing Method Based on Internet of Things. In: Proceedings of the International Conference on Intelligent Systems Research and Mechatronics Engineering (ISRME 2015), Zhengzhou, China, April 11-13, (2015)
47. Marinelli, P., Coen, C.S., Vitali, F.: SchemaPath, a Minimal Extension to XML Schema for Conditional Constraints. In: Proceedings of the 13th International World Wide Web Conference (WWW 2004), New York, USA, May 17-22, pp. 164-174 (2004)
48. MATLAB. [Online]. Available: http://uk.mathworks.com/products/matlab/index.html [Accessed: 23 March 2014]

49. Maximum Compression. [Online]. Available: http://www.maximumcompression.com/ [Accessed: 23 March 2014]
50. Min, J.-K., Park, M.-J., Chung, C.-W.: XPRESS: A Queriable Compression for XML Data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2003), San Diego, California, USA, June 09-12, pp. 122–133 (2003)
51. Müldner, T., Miziołek, J.K., Corbin, T.: Annotated Trees and their Applications to XML Compression. In: Proceedings of the 10th International Conference on Web Information Systems and Technologies (WEBIST 2014), Barcelona, Spain, April 03-05, pp. 27-39 (2014)
52. Ng, W., Lam, W.-Y., Wood, P.T., Levene, M.: XCQ: A Queriable XML Compression System. Journal of Knowledge and Information Systems. 10(4), 421–452 (2006)
53. Ng, W., Lau, H.L., Zhou, A.: Divide, Compress and Conquer: Querying XML via Partitioned Path-Based Compressed Data Blocks. World Wide Web – Internet and Web Information Systems. 11(2), 169-197 (2008)
54. N-Gram Extraction Tools. [Online]. Available: http://homepages.inf.ed.ac.uk/lzhang10/ngram.html [Accessed: 20 November 2013]
55. Notepad++. [Online]. Available: https://notepad-plus-plus.org/ [Accessed: 23 March 2014]
56. Nurseitov, N., Paulson, M., Reynolds, R., Izurieta, C.: Comparison of JSON and XML Data Interchange Formats: A Case Study. In Proceedings of the ISCA 22nd International Conference on Computer Applications in Industry and Engineering (CAINE 2009), San Francisco, California, USA, November 04-06, pp. 157–162 (2009)
57. O'Boyle, N.M.: Towards a Universal SMILES Representation – A Standard Method to Generate Canonical SMILES Based on the InChI. Journal of Cheminformatics. 4(22), (2012)
58. Oxygen XML Editor. [Online]. Available: http://www.oxygenxml.com/ [Accessed: 20 November 2013]

59. QSAR Model Reporting Format Inventory. [Online]. Available: http://qsardb.jrc.it/qmrf/index.jsp [Accessed: 15 October 2012]
60. Radescu, R.: Transform Methods used in Lossless Compression of Text Files. Romanian Journal of Information Science and Technology. 12(1), 101 – 115 (2009)
61. Rahmoun, A., Elberrichi, Z.: Experimenting N-Grams in Text Categorization. International Arab Journal of Information Technology, 4(4), 377-385 (2007)
62. Ratanaworabhan, P., Ke, J., Burtscher, M.: Fast Lossless Compression of Scientific Floating-Point Data. In: Proceedings of the IEEE Data Compression Conference (DCC 2006), Snowbird, Utah, USA, March 28-30, pp. 133–142 (2006)
63. rdfdata.org data. [Online]. Available: http://www.rdfdata.org/data.html [Accessed: 22 January 2012]
64. Rexline, S.J., Robert, L.: Dictionary Based Preprocessing Methods in Text Compression - A Survey. International Journal of Wisdom Based Computing. 1(2), 13-18 (2011)
65. RNAdb. [Online]. Available: http://research.imb.uq.edu.au/rnadb/ [Accessed: 22 January 2012]
66. Sakr, S.: XML Compression Techniques: A Survey and Comparison. Journal of Computer and System Sciences. 75(5), 303-322 (2009)
67. Scanlon, S., Ridley, M.: A Fully Reversible Data Transform Technique Enhancing Data Compression of SMILES Data. In: Proceedings of the IFIP WG 8.4, 8.9, TC 5 International Cross-Domain Availability, Reliability, and Security Conference in Information Systems and HCI (CD-ARES 2013), Regensburg, Bavaria, Germany, September 02-06, pp. 54-68 (2013)
68. Skibiński, P.: Reversible Data Transforms that Improve Effectiveness of Universal Lossless Data Compression. PhD Dissertation, University of Wrocław, Wrocław, Poland (2006)
69. Skibiński, P.: Two-Level Directory Based Compression. In: Proceedings of the IEEE Data Compression Conference (DCC 2005), Snowbird, Utah, USA, March 29-31, pp. 481–492 (2005)

70. Skibiński, P., Grabowski, S., Swacha, J.: Effective Asymmetric XML Compression. Software – Practice and Experience. 38(10), 1027-1047 (2008)
71. Skibiński, P., Grabowski, S., Swacha, J.: Fast Transform for Effective XML Compression. In: Proceedings of the 9th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics, Lviv, Ukraine, February 19-24, pp. 323-326 (2007)
72. Skibiński, P., Swacha, J.: Combining Efficient XML Compression with Query Processing. In: Proceedings of the 11th East European Conference on Advances in Databases and Information Systems, Varna, Bulgaria, September 29-October 03, 4690, pp. 330-342 (2007)
73. Soft 112 XMill. [Online]. Available: http://xmill.soft112.com [Accessed: 15 January 2016]
74. SPSS Software. [Online]. Available: http://www-01.ibm.com/software/uk/analytics/spss/ [Accessed: 15 January 2016]
75. The GZip Home Page. [Online]. Available: http://www.gzip.org/ [Accessed: 12 November 2012]
76. Timmerer, C., Kofler, I., Liegl, J., Hellwagner, H.: An Evaluation of Existing Metadata Compression and Encoding Technologies for MPEG-21 Applications. In: Proceedings of the 7th IEEE International Symposium on Multimedia, Irvine, California, USA, December 12-14, pp. 534-539 (2005)
77. ToxCast. [Online]. Available: http://www.epa.gov/ncct/toxcast/ [Accessed: 15 October 2012]
78. Tourwe, T., Stoops, L., Decneut, S.: Automated Support for Data Exchange via XML. In: Proceedings of the IEEE Fifth International Symposium on Multimedia Software Engineering (ISMSE 2003), Taichung, Taiwan, December 10-12, pp. 70-77 (2003)
79. Unicode.org, General Structure, Chapter 2. [Online]. Available: http://www.unicode.org/versions/Unicode8.0.0/ch02.pdf [Accessed: 10 June 2015]
80. Universal Binary JSON Specification. [Online]. Available: http://ubjson.org/

[Accessed: 10 June 2015]

81. Wang, H., Li, J., Luo, J., He, Z.: XCpaqs: Compression of XML Document with XPath Query Support. In: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2004), Las Vegas, Nevada, USA, April 05-07, pp. 354-358 (2004)
82. Weininger, D.: SMILES, A Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. Journal of Chemical Information and Computer Sciences. 28(1), 31–36 (1988)
83. Why XML? [Online]. Available: http://www.simonstl.com/articles/whyxml.htm [Accessed: 12 March 2014]
84. Wikimedia Downloads. [Online]. Available: http://dumps.wikimedia.org/ [Accessed: 22 January 2012]
85. XMLCompBench: Benchmark of XML Compression Tools. [Online]. Available: http://xmlcompbench.sourceforge.net/Dataset.html [Accessed: 22 January 2012]
86. XML Data Repository. [Online]. Available: http://www.cs.washington.edu/research/xmldatasets/www/repository.html [Accessed: 22 January 2012]
87. YAML Ain't Markup Language (YAML) Version 1.2. [Online]. Available: http://yaml.org/spec/cvs/spec.pdf [Accessed: 10 June 2015]
88. Zhang, S., Bao, X., Shu, J., Chen, S.: Comparative Research of XML Compression Technologies. In: Proceedings of The 9th International Conference for Young Computer Scientists, Hunan, China, November 18-21, pp. 112-117 (2008)
89. Zhang, X., Zhai, G., Tian, R.: XML Schema Based Compression Technology Over XML Data Stream. In: Proceedings of the 6th Conference on Web Information Systems and Applications (WISA '09), Xuzhou, China, September 18-20, pp. 27-31 (2009).
90. ZPAQ: Incremental Journaling Backup Utility and Archiver. [Online]. Available: http://mattmahoney.net/dc/zpaq.html [Accessed: 12 November 2012]

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
Auction	321gone	0.02	311	0	17409	5	0.69
Auction	Ebay	0.03	156	0	31539	5	0.8909
Auction	Ubid	0.02	342	0	12583	5	0.5941
Auction	Yahoo	0.03	342	0	17701	5	0.6733
Courses	Reed	0.3	10546	0	136883	4	0.4421
Courses	Rice	0.8	18116	0	509564	5	0.6089
Courses	Uwm	2.73	76132	6	1376010	5	0.4846
Courses	Washington	3.56	95337	0	1810278	6	0.4874
Courses	Wsu	1.78	74557	0	728715	4	0.3957
DBLP_SIGMOD	SigmodRecord	0.49	11526	3737	248725	6	0.5946
DSSTox	ARYEXP_Aux_v2a_2556_06Mar2009_nostructures	17.5	171253	0	10604418	3	0.5772
DSSTox	ARYEXP_v2a_958_06Mar2009_nostructures	1.65	24909	0	716849	3	0.4134
DSSTox	CPDBAS_v5d_1547_20Nov2008_nostructures	5.62	94368	0	1591073	3	0.2701
DSSTox	DBPCAN_v4b_209_15Feb2008_nostructures	0.4	6480	0	144140	3	0.3443
DSSTox	EPAFHM_v4b_617_15Feb2008_nostructures	1.18	23447	0	414276	3	0.3348
DSSTox	FDAMDD_v3b_1216_15Feb2008_nostructures	2.46	38913	0	1022762	3	0.3958
DSSTox	GEOGSE_Aux_v2a_2700_09Mar2009_nostructures	15.6	110701	0	11544553	3	0.7055
DSSTox	GEOGSE_v2a_1179_09Mar2009_nostructures	2.03	30655	0	875015	3	0.4114
DSSTox	HPVCSI_v2c_3548_15Feb2008_nostructures	5.7	88701	0	2112578	3	0.3535
DSSTox	HPVISD_v1b_1006_15Feb2008_nostructures	1.31	21127	0	511984	3	0.372
DSSTox	IRISTR_v1b_544_15Feb2008_nostructures	2.2	30465	0	709507	3	0.3091

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
DSSTox	ISSCAN_v3a_1153_19Sept08	1.13	24214	0	548747	3	0.4618
DSSTox	KIERBL_v1a_278_17Feb2009_nostructures	0.56	9453	0	211471	3	0.3629
DSSTox	NCTRER_v4b_232_15Feb2008_nostructures	0.52	8817	0	204761	3	0.3784
DSSTox	NTPBSI_v4c_2330_04Aug2009_nostructures	4.08	65241	0	1516553	3	0.3565
DSSTox	NTPHTS_v2b_1408_15Feb2008_nostructures	1.81	30977	0	663957	3	0.3499
DSSTox	TOX21S_v2a_8193_22Mar2012_nostructures	11.1	172054	0	4444729	3	0.3817
DSSTox	TOXCST_v4a_1892_20Mar2012_nostructures	3.17	54869	0	1081530	3	0.3258
EXI	A-TIGR-1-ArrayDesign	22	226523	226550	6123488	10	0.7536
EXI	A-TIGR-1-BioSequence	43.1	567523	441421	12107080	8	0.5984
EXI	A-TIGR-1-DesignElement	68	758837	442165	13290183	8	0.5674
EXI	Factbook	4.22	55453	0	2588411	5	0.6168
EXI	inv1	0.01	90	73	1684	7	0.5979
EXI	inv10	0.01	225	199	4047	7	0.5938
EXI	inv100	0.1	1575	1459	27682	7	0.592
EXI	inv1000	0.96	15075	14059	263492	7	0.5919
EXI	inv50	0.05	825	759	14615	7	0.5944
EXI	inv500	0.48	7575	7059	131855	7	0.5914
EXI	Iron_array	1.06	16754	11880	371535	10	0.545
EXI	Periodic	0.11	1897	936	30833	3	0.4805
EXI	telemCompTest10M	10.4	177634	414370	691256	7	0.6855
EXI	telemCompTest1M	1.03	16760	39032	65484	7	0.6999

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
EXI	Weblog	2.78	93435	0	1093675	3	0.3746
ExpoCast	AHHS	7.07	139250	0	4259664	3	0.5782
ExpoCast	CCC	5.65	156509	0	2366298	3	0.4064
ExpoCast	CTEPP_NC	28.1	746406	0	12532868	3	0.4306
ExpoCast	CTEPP_OH	28.4	757131	0	12682262	3	0.43
NASA	Nasa	25.1	476646	53882	16377956	8	0.6827
PSD_SwissProt	SwissProt	118	2977031	2189859	45636090	5	0.6332
QSAR	download_qmrf108	0.02	119	376	8812	5	0.8711
QSAR	download_qmrf112	0.03	135	422	11759	6	0.8925
QSAR	download_qmrf119	0.03	119	364	12647	6	0.886
QSAR	download_qmrf121	0.03	117	357	12700	6	0.8872
QSAR	download_qmrf126	0.03	129	412	11335	6	0.88
QSAR	download_qmrf132	0.03	105	325	15305	5	0.8983
QSAR	download_qmrf135	0.03	135	442	11969	6	0.8909
QSAR	download_qmrf136	0.03	141	462	12680	6	0.883
QSAR	download_qmrf140	0.03	123	385	11027	6	0.8783
QSAR	download_qmrf143	0.03	137	424	15031	6	0.8965
QSAR	download_qmrf144	0.03	117	357	12277	6	0.8858
QSAR	download_qmrf150	0.03	113	346	12090	6	0.8821
QSAR	download_qmrf153	0.03	141	460	9354	6	0.867
QSAR	download_qmrf154	0.02	103	318	7313	5	0.8612

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
QSAR	download_qmrf155	0.02	119	367	7064	6	0.8578
QSAR	download_qmrf158	0.03	145	452	11857	6	0.902
QSAR	download_qmrf160	0.03	137	424	11001	6	0.8972
QSAR	download_qmrf162	0.02	110	331	8170	6	0.8715
QSAR	download_qmrf169	0.03	119	364	12103	6	0.8837
QSAR	download_qmrf171	0.03	141	461	11401	6	0.8785
QSAR	download_qmrf173	0.03	149	483	11083	6	0.8839
QSAR	download_qmrf174	0.03	141	461	10495	6	0.8746
QSAR	download_qmrf175	0.03	137	448	10166	6	0.8724
QSAR	download_qmrf176	0.03	113	345	10985	6	0.8816
QSAR	download_qmrf177	0.03	145	474	13039	6	0.8901
QSAR	download_qmrf179	0.03	149	484	12295	6	0.8921
QSAR	download_qmrf184	0.03	137	449	12216	6	0.8861
QSAR	download_qmrf207	0.03	133	436	10319	6	0.8816
QSAR	download_qmrf208	0.03	137	448	11005	6	0.8833
QSAR	download_qmrf209	0.03	137	448	12071	6	0.8887
QSAR	download_qmrf220	0.03	137	449	12865	6	0.8904
QSAR	download_qmrf221	0.03	135	442	11835	6	0.8908
QSAR	download_qmrf222	0.03	135	441	11664	6	0.8907
QSAR	download_qmrf224	0.03	137	448	12322	6	0.8928
QSAR	download_qmrf225	0.03	160	527	14060	6	0.8949

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
QSAR	download_qmrf226	0.03	150	497	12611	6	0.8896
QSAR	download_qmrf241	0.04	147	483	22687	6	0.918
QSAR	download_qmrf242	0.04	147	460	16174	6	0.9
QSAR	download_qmrf245	0.02	107	328	8277	6	0.8596
QSAR	download_qmrf264	0.04	153	498	15573	6	0.9016
QSAR	download_qmrf265	0.03	147	480	11656	6	0.8859
QSAR	download_qmrf266	0.03	145	473	11761	6	0.8991
QSAR	download_qmrf267	0.03	148	489	12890	6	0.8926
QSAR	download_qmrf288	0.03	135	443	11192	6	0.8844
QSAR	download_qmrf289	0.04	164	533	19453	6	0.9129
QSAR	download_qmrf290	0.03	129	398	12091	6	0.8967
QSAR	download_qmrf291	0.03	135	420	13046	6	0.902
QSAR	download_qmrf292	0.03	131	404	11349	6	0.8939
QSAR	download_qmrf295	0.02	107	336	5784	5	0.8528
QSAR	download_qmrf296	0.02	113	357	5376	5	0.8495
QSAR	download_qmrf297	0.03	142	461	11058	6	0.8894
QSAR	download_qmrf299	0.03	139	455	14663	6	0.9068
QSAR	download_qmrf300	0.03	141	461	12650	6	0.8904
QSAR	download_qmrf303	0.03	139	433	12750	6	0.91
QSAR	download_qmrf309	0.03	127	407	13168	6	0.8889
QSAR	download_qmrf311	0.04	164	536	19120	6	0.9142

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
QSAR	download_qmrf312	0.03	105	320	11224	5	0.885
QSAR	download_qmrf313	0.03	107	331	12280	5	0.89
QSAR	download_qmrf314	0.03	103	315	11663	5	0.8882
QSAR	download_qmrf315	0.03	109	341	15014	5	0.9021
QSAR	download_qmrf317	0.02	107	334	6101	6	0.848
QSAR	download_qmrf318	0.02	109	340	6029	6	0.8466
QSAR	download_qmrf319	0.02	105	328	6657	6	0.8514
QSAR	download_qmrf320	0.02	107	334	6119	6	0.847
QSAR	download_qmrf331	0.04	131	419	21665	6	0.9135
QSAR	download_qmrf332	0.03	154	503	13241	6	0.8932
QSAR	download_qmrf333	0.03	150	486	11020	6	0.8847
QSAR	download_qmrf336	0.03	154	508	12726	6	0.8944
QSAR	download_qmrf53	0.02	115	356	9443	5	0.8714
QSAR	download_qmrf83	0.03	134	435	13238	6	0.8945
QSAR	qmrf143_AAT_FHM_ANNE_Supporting_Data_allcompounds	0.4	11541	0	202498	3	0.4841
QSAR	qmrf143_AAT_FHM_ANNE_Supporting_Data_dsstoxdb	1.27	30582	0	428554	3	0.3213
QSAR	qmrf155_HAA_training_40	0.02	561	0	8244	3	0.4113
QSAR	qmrf162_Test	0.01	181	0	2298	3	0.4173
QSAR	qmrf162_Training	0.05	1596	0	19636	3	0.3998
RDF	_corpdata	0.18	3053	2852	15863	6	0.7269

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
RDF	_out	0.84	12408	9702	245208	7	0.7231
RDF	agenda_62	0.08	1389	318	32889	6	0.5084
RDF	allpolls	1.08	19715	10003	229696	3	0.5079
RDF	bib1300	0.62	10483	1310	313307	3	0.5637
RDF	css3deps	0.01	97	96	647	3	0.8045
RDF	cwm-crawler-output	1.06	12854	9046	226223	7	0.7428
RDF	cyc	23.8	255054	151860	14769765	3	-
RDF	dc	0.14	1839	1411	45820	7	0.6703
RDF	e164	0.04	637	260	6138	5	0.4594
RDF	ins	0.43	4764	2897	242215	5	0.8087
RDF	mrftodoaddenda.d	0.08	1000	476	38103	3	0.7356
RDF	news	2.6	3814	1	2414859	4	0.9757
RDF	public-groups	0.13	2087	1349	36794	3	0.6617
RDF	rdf-result	0.11	1403	1401	16605	3	0.8231
RDF	sample-publications	0.05	953	131	32761	3	0.6801
RDF	sneeair	5.54	111087	20572	1494053	3	0.4625
RDF	sw_community	0.1	1172	1084	32525	9	0.7916
RDF	TheMatrix	0.15	2115	1452	38903	3	0.7813
RDF	tools	0.09	1696	388	51809	3	0.673
RDF	tr	0.8	11571	7580	239881	4	0.6969
RDF	UNSPSC	1.84	29388	19591	515349	3	0.6545

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
RDF	ww-validation-results	0.15	1871	1559	7488	4	0.731
RNA	asloverlaps_hg18	3.28	63869	0	1190951	3	0.3465
RNA	asloverlaps_mm8	4.78	97637	0	1580688	3	0.3151
RNA	asoverlapseqs_hg18	2.18	9613	0	2012704	3	0.8792
RNA	asoverlapseqs_mm8	3.26	14536	0	2999581	3	0.8776
RNA	tblCombinedLit	2.52	23692	2	1931041	3	0.733
RNA	tblEvofold	19.5	427582	2	9289155	3	0.4542
RNA	tblHinv	1.61	28344	2	933859	3	0.5538
RNA	tblmiRNA	1.09	20262	2	628784	3	0.5507
RNA	tblNcrnascan	1.29	22726	2	744792	3	0.5524
RNA	tblpiRNA	94.5	2100400	2	46206685	3	0.4662
RNA	tblRnaz	18.1	323857	2	10750852	3	0.5661
RNA	tblsnoRNA	0.44	9047	2	211751	3	0.4555
RNA	XMLFantom3	82.8	477939	2	74138025	3	0.8538
Shakespeare	all_shakes	7.78	179690	0	5449539	7	0.6725
Shakespeare	half_shakes	5.11	117535	0	3586575	7	0.674
Shakespeare	third_shakes	3.31	75432	0	2341119	7	0.6783
ToxCast	ACEA_PrimaryData_ToxCast-PhaseI_320Chemicals_03Feb2010	239	6635521	0	100947992	3	0.4028
ToxCast	Attagene_PrimaryData_ToxCast-PhaseI_320Chemicals-plus-controls_29Jan2010	54.3	1453121	0	22445567	3	0.3941

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
ToxCast	Bioseek_PrimaryData_ToxCast-PhaseI_320Chemicals_03Feb2010	19.7	524281	0	8150900	3	0.3956
ToxCast	Cellumen_PrimaryData_ToxCast-PhaseI_320Chemicals-plus-controls_03Feb2010	283	8225281	0	117807524	3	0.3975
ToxCast	CellzDirect_PrimaryData_ToxCast-PhaseI_320Chemicals_29Jan2010	167	5446376	0	61642007	3	0.3528
ToxCast	Chemicals_Sample_Map_20091214	0.09	2561	0	37053	3	0.3861
ToxCast	Gentronix_PrimaryData_ToxCast-PhaseI_320Chemicals_14Dec2009	0.1	2561	0	45629	3	0.4286
ToxCast	landscape_assay_coverage_20080908_SUPPLEMENTAL	37	872433	0	6598941	3	0.1703
ToxCast	landscape_assays_SUPPLEMENTAL_biochemical	0.79	6949	0	677456	3	0.8205
ToxCast	landscape_assays_SUPPLEMENTAL_human_metabolome	0	15	0	451	3	0.5628
ToxCast	landscape_assays_SUPPLEMENTAL_regulations	0.04	295	0	36913	3	0.8538
ToxCast	landscape_assays_SUPPLEMENTAL_toxicology	0.12	1373	0	96537	3	0.7628
ToxCast	landscape_data_sources_SUPPLEMENTAL_assay_data	0.03	349	0	18778	3	0.6574
ToxCast	landscape_data_sources_SUPPLEMENTAL_regulatory	0.04	289	0	37754	3	0.8231
ToxCast	landscape_data_sources_SUPPLEMENTAL_screening	0.02	141	0	12233	3	0.736

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
ToxCast	NCGC_PrimaryData_ToxCast-PhaseI_320Chemicals_03Feb2010	23.4	589816	0	9615369	3	0.3926
ToxCast	Phase_1_ACEA_20110110	0.11	3400	0	47337	3	0.4093
ToxCast	Phase_1_Attagene_20110110	1.04	26266	0	319638	3	0.2918
ToxCast	Phase_1_BioSeek_20110110	2.79	55003	0	704858	3	0.2405
ToxCast	Phase_1_Cellumen_20110110	1.09	18850	0	240405	3	0.2107
ToxCast	Phase_1_CellzDirect_20110110	0.66	16069	0	197029	3	0.286
ToxCast	Phase_1_ChemClass_20110110	1.25	33064	0	238487	3	0.1816
ToxCast	Phase_1_Chemicals_20110110	0.05	1546	0	30125	3	0.5851
ToxCast	Phase_1_EPISuite_20110110	0.12	3709	0	47962	3	0.3867
ToxCast	Phase_1_Gentronix_20110110	0.04	1546	0	19575	3	0.4446
ToxCast	Phase_1_LeadScope_20110110	0.08	3091	0	34537	3	0.3882
ToxCast	Phase_1_MN_MetabogenRxnClass_20110110	3.06	39862	0	286142	3	0.0892
ToxCast	Phase_1_MN_WholeMolecProperties_20110110	0.19	5872	0	60807	3	0.3075
ToxCast	Phase_1_NCGC_20110110	0.33	7108	0	91416	3	0.2682
ToxCast	Phase_1_Novascreen_20110110	3.71	79414	0	1024901	3	0.2632
ToxCast	Phase_1_PhysChem_derived_20110110	0.05	1855	0	27570	3	0.5031
ToxCast	Phase_1_QikProp_20110110	0.54	16378	0	163140	3	0.2899
ToxCast	Phase_1_Solidus_20110110	0.09	2473	0	30610	3	0.3402
ToxCast	Phase_1_StructureClassifiers_20110110	3.76	76324	0	541925	3	0.1374
ToxCast	Phase_1_ToxRefDB_20110110	6.59	79414	0	932690	3	0.1349

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
ToxCast	Rotroff_et_al_supplemental_data_file_2	0.27	11341	0	138473	3	0.4817
ToxCast	Rotroff_et_all_supplemental_data_file_1	11.8	314161	0	3770176	3	0.3049
ToxCast	Supplemental_File_1	2.06	62349	0	725248	3	0.3359
ToxCast	Supplemental_File_3	3.51	91981	0	1399079	3	0.3803
ToxCast	Supplemental_File_4	0.03	1217	0	10188	3	0.3065
ToxCast	Supplemental_File_5	0.01	533	0	4784	3	0.3212
ToxCast	Table_S1_ToxCastAssayMaster_20091214	2.79	55966	0	789937	3	0.2706
ToxCast	Table_S2_Hits_under_1uM	0.13	4141	0	56719	3	0.419
ToxCast	Table_S3_S4_Pathway_assay_results_tableS3	3.84	40126	0	550457	3	0.1369
ToxCast	Table_S3_S4_Pathway_assay_results_tableS4	0.04	449	0	4773	3	0.1131
ToxCast	TableS1_AssayDescriptions_cis	0.01	289	0	4876	3	0.3382
ToxCast	TableS1_AssayDescriptions_trans	0.01	126	0	2158	3	0.3979
ToxCast	TableS2_ConcResp_AC50Data	29.4	1063541	0	13058255	3	0.4243
ToxCast	TableS3_RelativeRiskPermutationTest	0.95	27741	0	387767	3	0.3907
ToxCast	ToxCastAssayMaster_20091214	2.79	55966	0	789937	3	0.2706
ToxCast	ToxCastAssayMaster_20100128	2.94	58969	0	835184	3	0.2713
ToxCast	ToxCastP1_320_ChemicalQC_15Dec2009_bins	0.13	2881	0	40030	3	0.305
TPC-H	customer	0.54	13501	1	280129	3	0.4944
TPC-H	lineitem	34.6	1022976	1	10331569	3	0.2849
TPC-H	nation	0	126	1	2480	3	0.4958
TPC-H	orders	5.67	150001	1	2108816	3	0.3545

Appendix A: XML Data Characteristics

Data Group	File_Name	Size (MB)	Elements	Attributes	Plain Text Characters	Depth	Structure - Data Ratio
TPC-H	part	0.66	20001	1	298154	3	0.4295
TPC-H	partsupp	2.31	48001	1	1297837	3	0.5368
TPC-H	region	0	21	1	458	3	0.5531
TPC-H	supplier	0.03	801	1	16019	3	0.4973
Treebank	treebank_e	88.1	2437666	1	66434706	36	-
Wikimedia	enwikibooks-20120325-pages-articles	390	1235069	106443	371644185	5	0.9564
Wikimedia	enwikinews-20120325-pages-articles	155	1077844	93065	138304634	5	0.8966
Wikimedia	enwikiquote-20120324-pages-articles	217	548958	53373	212358029	5	0.9653
Wikimedia	enwikiversity-20120320-pages-articles	176	747527	63360	163310710	5	0.9412
XBench	XBench-DCSD-Normal	110	2242699	150000	53086067	8	0.4815
XBench	XBench-DCSD-Small	11	225292	15000	5325670	8	0.4809
XBench	XBench-TCSD-Normal	112	2749751	7333	93554073	8	0.801
XBench	XBench-TCSD-Small	17.2	-	-	-	-	0.9575
XBench	XMark1	11.5	167865	38265	8568932	12	0.7783
XBench	XMark2	115	1666315	381878	85298671	12	0.7795

XML statistics could not be obtained for the XBench-TCSD-Small file due to a processing error. XML structure-data ratios could not be computed for the cyc file in the RDF data group and the treebank_e file due to an issue with removing the structure of the documents.

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
Auction	321gone	0.02	0.8003
Auction	ebay	0.03	0.9417
Auction	ubid	0.01	0.6966
Auction	yahoo	0.02	0.7814
Courses	reed	0.17	0.5106
Courses	rice	0.54	0.7185
Courses	uwm	1.48	0.56
Courses	washington	1.8	0.5058
Courses	wsu	1.04	0.4757
DBLP_SIGMOD	SigmodRecord	0.36	0.8012
DSSTox	ARYEXP_Aux_v2a_2556_06Mar2009_nostructures	13.88	0.7282
DSSTox	ARYEXP_v2a_958_06Mar2009_nostructures	1.15	0.5729
DSSTox	CPDBAS_v5d_1547_20Nov2008_nostructures	3.5	0.4038
DSSTox	DBPCAN_v4b_209_15Feb2008_nostructures	0.26	0.4936
DSSTox	EPAFHM_v4b_617_15Feb2008_nostructures	0.76	0.4741
DSSTox	FDAMDD_v3b_1216_15Feb2008_nostructures	1.68	0.5503
DSSTox	GEOGSE_Aux_v2a_2700_09Mar2009_nostructures	13.46	0.8263
DSSTox	GEOGSE_v2a_1179_09Mar2009_nostructures	1.41	0.5708
DSSTox	HPVCSI_v2c_3548_15Feb2008_nostructures	3.78	0.5071
DSSTox	HPVISD_v1b_1006_15Feb2008_nostructures	0.88	0.5276
DSSTox	IRISTR_v1b_544_15Feb2008_nostructures	1.41	0.4523
DSSTox	ISSCAN_v3a_1153_19Sept08	0.78	0.6036

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
DSSTox	KIERBL_v1a_278_17Feb2009_nostructures	0.37	0.5144
DSSTox	NCTRER_v4b_232_15Feb2008_nostructures	0.34	0.5262
DSSTox	NTPBSI_v4c_2330_04Aug2009_nostructures	2.68	0.5038
DSSTox	NTPHTS_v2b_1408_15Feb2008_nostructures	1.19	0.4981
DSSTox	TOX21S_v2a_8193_22Mar2012_nostructures	7.5	0.5377
DSSTox	TOXCST_v4a_1892_20Mar2012_nostructures	2.04	0.4698
EXI	A-TIGR-1-ArrayDesign	12.96	0.8384
EXI	A-TIGR-1-BioSequence	23.37	0.6347
EXI	A-TIGR-1-DesignElement	46.61	0.6751
EXI	factbook	3.5	0.7268
EXI	inv1	0	0.7561
EXI	inv10	0.01	0.7641
EXI	inv100	0.09	0.7685
EXI	inv1000	0.83	0.7692
EXI	inv50	0.05	0.7692
EXI	inv500	0.42	0.7687
EXI	Iron_array	0.55	0.561
EXI	periodic	0.08	0.6333
EXI	telemCompTest10M	9.74	0.7526
EXI	telemCompTest1M	0.98	0.7682
EXI	weblog	2.03	0.58
ExpoCast	AHHS	5.3	0.7197

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
ExpoCast	CCC	3.66	0.5437
ExpoCast	CTEPP_NC	18.77	0.576
ExpoCast	CTEPP_OH	19	0.5751
NASA	nasa	18.62	0.8239
PSD_SwissProt	SwissProt	87.41	0.873
QSAR	download_qmrf108	0.02	0.9234
QSAR	download_qmrf112	0.03	0.9316
QSAR	download_qmrf119	0.03	0.9311
QSAR	download_qmrf121	0.03	0.9313
QSAR	download_qmrf126	0.03	0.9273
QSAR	download_qmrf132	0.03	0.9361
QSAR	download_qmrf135	0.03	0.9342
QSAR	download_qmrf136	0.03	0.9323
QSAR	download_qmrf140	0.03	0.9248
QSAR	download_qmrf143	0.03	0.9377
QSAR	download_qmrf144	0.03	0.9301
QSAR	download_qmrf150	0.03	0.9267
QSAR	download_qmrf153	0.02	0.9244
QSAR	download_qmrf154	0.02	0.9118
QSAR	download_qmrf155	0.02	0.9141
QSAR	download_qmrf158	0.03	0.93
QSAR	download_qmrf160	0.03	0.9298

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
QSAR	download_qmrf162	0.02	0.9118
QSAR	download_qmrf169	0.03	0.9284
QSAR	download_qmrf171	0.03	0.9289
QSAR	download_qmrf173	0.03	0.9351
QSAR	download_qmrf174	0.03	0.927
QSAR	download_qmrf175	0.03	0.9245
QSAR	download_qmrf176	0.03	0.9253
QSAR	download_qmrf177	0.03	0.9359
QSAR	download_qmrf179	0.03	0.9404
QSAR	download_qmrf184	0.03	0.9315
QSAR	download_qmrf207	0.03	0.9287
QSAR	download_qmrf208	0.03	0.9286
QSAR	download_qmrf209	0.03	0.9317
QSAR	download_qmrf220	0.03	0.9327
QSAR	download_qmrf221	0.03	0.9312
QSAR	download_qmrf222	0.03	0.9313
QSAR	download_qmrf224	0.03	0.933
QSAR	download_qmrf225	0.03	0.9407
QSAR	download_qmrf226	0.03	0.9367
QSAR	download_qmrf241	0.04	0.952
QSAR	download_qmrf242	0.03	0.9451
QSAR	download_qmrf245	0.02	0.9135

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
QSAR	download_qmrf264	0.03	0.941
QSAR	download_qmrf265	0.03	0.9336
QSAR	download_qmrf266	0.03	0.9327
QSAR	download_qmrf267	0.03	0.9364
QSAR	download_qmrf288	0.03	0.9274
QSAR	download_qmrf289	0.04	0.9503
QSAR	download_qmrf290	0.03	0.9356
QSAR	download_qmrf291	0.03	0.939
QSAR	download_qmrf292	0.03	0.934
QSAR	download_qmrf295	0.02	0.9069
QSAR	download_qmrf296	0.02	0.9077
QSAR	download_qmrf297	0.03	0.9332
QSAR	download_qmrf299	0.03	0.9369
QSAR	download_qmrf300	0.03	0.9362
QSAR	download_qmrf303	0.03	0.9378
QSAR	download_qmrf309	0.03	0.9349
QSAR	download_qmrf311	0.04	0.9493
QSAR	download_qmrf312	0.02	0.9306
QSAR	download_qmrf313	0.03	0.9299
QSAR	download_qmrf314	0.02	0.9277
QSAR	download_qmrf315	0.03	0.9376
QSAR	download_qmrf317	0.02	0.9004

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
QSAR	download_qmrf318	0.02	0.8999
QSAR	download_qmrf319	0.02	0.9033
QSAR	download_qmrf320	0.02	0.9002
QSAR	download_qmrf331	0.04	0.9507
QSAR	download_qmrf332	0.03	0.9361
QSAR	download_qmrf333	0.03	0.9317
QSAR	download_qmrf336	0.03	0.9364
QSAR	download_qmrf53	0.02	0.9228
QSAR	download_qmrf83	0.03	0.936
QSAR	qmrf143_AAT_FHM_ANNE_Supporting_Data_allcompounds	0.27	0.613
QSAR	qmrf143_AAT_FHM_ANNE_Supporting_Data_dsstoxdb	0.81	0.4591
QSAR	qmrf155_HAA_training_40	0.01	0.5342
QSAR	qmrf162_Test	0	0.5496
QSAR	qmrf162_Training	0.03	0.5219
RDF	_corpdata	0.15	0.9127
RDF	_out	0.61	0.9993
RDF	agenda_62	0.04	1
RDF	allpolls	0.77	1
RDF	bib1300	0.42	0.9999
RDF	css3deps	0.01	1.0003
RDF	cwm-crawler-output	0.85	0.9999
RDF	cyc	19.77	-

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
RDF	dc	0.1	1
RDF	e164	-	-
RDF	ins	0.4	1
RDF	mrftodoaddenda.d	0.07	1
RDF	news	2.6	0.9891
RDF	public-groups	0.1	0.9511
RDF	rdf-result	0.11	1
RDF	sample-publications	0.04	0.8288
RDF	sneeair	3.88	1
RDF	sw_community	0.09	0.9101
RDF	TheMatrix	0.13	0.9163
RDF	tools	0.07	0.8103
RDF	tr	0.61	0.9677
RDF	UNSPSC	1.24	0.8946
RDF	ww-validation-results	0.15	1
RNA	asloverlaps_hg18	2.27	0.5407
RNA	asloverlaps_mm8	3.21	0.5043
RNA	asoverlapseqs_hg18	2.05	0.9418
RNA	asoverlapseqs_mm8	3.06	0.9411
RNA	tblCombinedLit	2.19	0.8485
RNA	tblEvofold	13.89	0.6444
RNA	tblHinv	1.25	0.7168

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
RNA	tblmiRNA	0.84	0.7216
RNA	tblNcrnascan	0.98	0.7315
RNA	tblpiRNA	69.12	0.6473
RNA	tblRnaz	14.06	0.734
RNA	tblsnoRNA	0.32	0.6349
RNA	XMLFantom3	76.86	0.9238
Shakespeare	all_shakes	5.85	0.9061
Shakespeare	half_shakes	3.85	0.907
Shakespeare	third_shakes	2.5	0.9108
ToxCast	ACEA_PrimaryData_ToxCast-PhaseI_320Chemicals_03Feb2010	153.95	0.5433
ToxCast	Attagene_PrimaryData_ToxCast-PhaseI_320Chemicals-plus-controls_29Jan2010	34.92	0.5387
ToxCast	Bioseek_PrimaryData_ToxCast-PhaseI_320Chemicals_03Feb2010	12.65	0.5405
ToxCast	Cellumen_PrimaryData_ToxCast-PhaseI_320Chemicals-plus-controls_03Feb2010	183.28	0.5292
ToxCast	CellzDirect_PrimaryData_ToxCast-PhaseI_320Chemicals_29Jan2010	101.88	0.4632
ToxCast	Chemicals_Sample_Map_20091214	0.06	0.5332
ToxCast	Gentronix_PrimaryData_ToxCast-PhaseI_320Chemicals_14Dec2009	0.07	0.5784
ToxCast	landscape_assay_coverage_20080908_SUPPLEMENTAL	19.31	0.1998
ToxCast	landscape_assays_SUPPLEMENTAL_biochemical	0.71	0.905
ToxCast	landscape_assays_SUPPLEMENTAL_human_metabolome	0	0.7359

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
ToxCast	landscape_assays_SUPPLEMENTAL_regulations	0.04	0.9243
ToxCast	landscape_assays_SUPPLEMENTAL_toxicology	0.1	0.8674
ToxCast	landscape_data_sources_SUPPLEMENTAL_assay_data	0.02	0.795
ToxCast	landscape_data_sources_SUPPLEMENTAL_regulatory	0.04	0.9058
ToxCast	landscape_data_sources_SUPPLEMENTAL_screening	0.01	0.8524
ToxCast	NCGC_PrimaryData_ToxCast-PhaseI_320Chemicals_03Feb2010	15.17	0.5386
ToxCast	Phase_1_ACEA_20110110	0.07	0.528
ToxCast	Phase_1_Attagene_20110110	0.62	0.382
ToxCast	Phase_1_BioSeek_20110110	1.6	0.3221
ToxCast	Phase_1_Cellumen_20110110	0.61	0.2875
ToxCast	Phase_1_CellzDirect_20110110	0.39	0.3784
ToxCast	Phase_1_ChemClass_20110110	0.66	0.2026
ToxCast	Phase_1_Chemicals_20110110	0.03	0.7383
ToxCast	Phase_1_EPISuite_20110110	0.07	0.4818
ToxCast	Phase_1_Gentronix_20110110	0.03	0.5879
ToxCast	Phase_1_LeadScope_20110110	0.05	0.4936
ToxCast	Phase_1_MN_MetabogenRxnClass_20110110	1.57	0.1017
ToxCast	Phase_1_MN_WholeMolecProperties_20110110	0.11	0.3888
ToxCast	Phase_1_NCGC_20110110	0.19	0.3598
ToxCast	Phase_1_Novascreen_20110110	2.16	0.3485
ToxCast	Phase_1_PhysChem_derived_20110110	0.03	0.6437
ToxCast	Phase_1_QikProp_20110110	0.31	0.3619

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
ToxCast	Phase_1_Solidus_20110110	0.05	0.4538
ToxCast	Phase_1_StructureClassifiers_20110110	1.94	0.1549
ToxCast	Phase_1_ToxRefDB_20110110	3.58	0.1944
ToxCast	Rotroff_et_al_supplemental_data_file_2	0.18	0.5892
ToxCast	Rotroff_et_all_supplemental_data_file_1	7.12	0.4143
ToxCast	Supplemental_File_1	1.26	0.4594
ToxCast	Supplemental_File_3	2.29	0.5269
ToxCast	Supplemental_File_4	0.02	0.3719
ToxCast	Supplemental_File_5	0.01	0.3945
ToxCast	Table_S1_ToxCastAssayMaster_20091214	1.73	0.4013
ToxCast	Table_S2_Hits_under_1uM	0.08	0.5707
ToxCast	Table_S3_S4_Pathway_assay_results_tableS3	2.15	0.2219
ToxCast	Table_S3_S4_Pathway_assay_results_tableS4	0.02	0.1745
ToxCast	TableS1_AssayDescriptions_cis	0.01	0.4875
ToxCast	TableS1_AssayDescriptions_trans	0	0.555
ToxCast	TableS2_ConcResp_AC50Data	18.76	0.5305
ToxCast	TableS3_RelativeRiskPermutationTest	0.58	0.5339
ToxCast	ToxCastAssayMaster_20091214	1.73	0.4013
ToxCast	ToxCastAssayMaster_20100128	1.82	0.4022
ToxCast	ToxCastP1_320_ChemicalQC_15Dec2009_bins	0.08	0.4317
TPC-H	customer	0.41	0.661
TPC-H	lineitem	22.3	0.4339

Appendix B: JSON Data Characteristics

Data Group	File Name	Original Size (MB)	Structure-Data Ratio
TPC-H	nation	0	0.6659
TPC-H	orders	3.84	0.5196
TPC-H	part	0.48	0.5999
TPC-H	partsupp	1.73	0.6953
TPC-H	region	0	0.7218
TPC-H	supplier	0.02	0.6633
Treebank	treebank_e	53.73	-
Wikimedia	enwikibooks-20120325-pages-articles	-	-
Wikimedia	enwikinews-20120325-pages-articles	143.41	0.9398
Wikimedia	enwikiquote-20120324-pages-articles	-	-
Wikimedia	enwikiversity-20120320-pages-articles	167.24	0.9669
XBench	XBench-DCSD-Normal	83.12	0.6599
XBench	XBench-DCSD-Small	8.34	0.6593
XBench	XBench-TCSD-Normal	102.37	0.8977
XBench	XBench-TCSD-Small	10.56	0.8981
XBench	XMark1	10.42	0.858
XBench	XMark2	104.1	0.8587

Files e164 from the RDF data group, enwikibooks and enwikiquote from the Wikimedia data group were not used in the JSON experiments due to transform issues during the conversion of XML to JSON. JSON structure-data ratios could not be computed for the cyc file in the RDF data group and the treebank_e file due to an issue with removing the structure of the documents.

Appendix C: ToxCast Missing Files

The following files are not included in the XML experiments. The affected files are for 1 to 5 Word-Ngram transforms used with the Existing_Numbers transform variation, except for the TableS3_RelativeRiskPermutationTest file, where 1 Word-Ngram was included in the experiments for this file:

- TableS3_RelativeRiskPermutationTest
- Table_S1_ToxCastAssayMaster_20091214
- Table_S2_Hits_under_1uM
- Table_S3_S4_Pathway_assay_results_tableS3
- Table_S3_S4_Pathway_assay_results_tableS4
- ToxCastAssayMaster_20091214
- ToxCastAssayMaster_20100128
- ToxCastP1_320_ChemicalQC_15Dec2009_bins
- Chemicals_Sample_Map_20091214
- Phase_1_ACEA_20110110
- Phase_1_Attagene_20110110
- Phase_1_BioSeek_20110110
- Phase_1_Cellumen_20110110
- Phase_1_CellzDirect_20110110
- Phase_1_ChemClass_20110110
- Phase_1_Chemicals_20110110
- Phase_1_EPISuite_20110110
- Phase_1_Gentronix_20110110
- Phase_1_LeadScope_20110110
- Phase_1_MN_MetabogenRxnClass_20110110
- Phase_1_MN_WholeMolecProperties_20110110
- Phase_1_NCGC_20110110
- Phase_1_Novascreen_20110110
- Phase_1_PhysChem_derived_20110110
- Phase_1_QikProp_20110110
- Phase_1_Solidus_20110110
- Phase_1_StructureClassifiers_20110110
- Phase_1_ToxRefDB_20110110

Appendix D: XML and JSON Overall Result Graphs and Tables

Figure 1. Overall Average Compression Ratios for XML AND JSON Compression Algorithms

Figure 2. Overall Average Compression Ratios for XML AND JSON Data Transforms

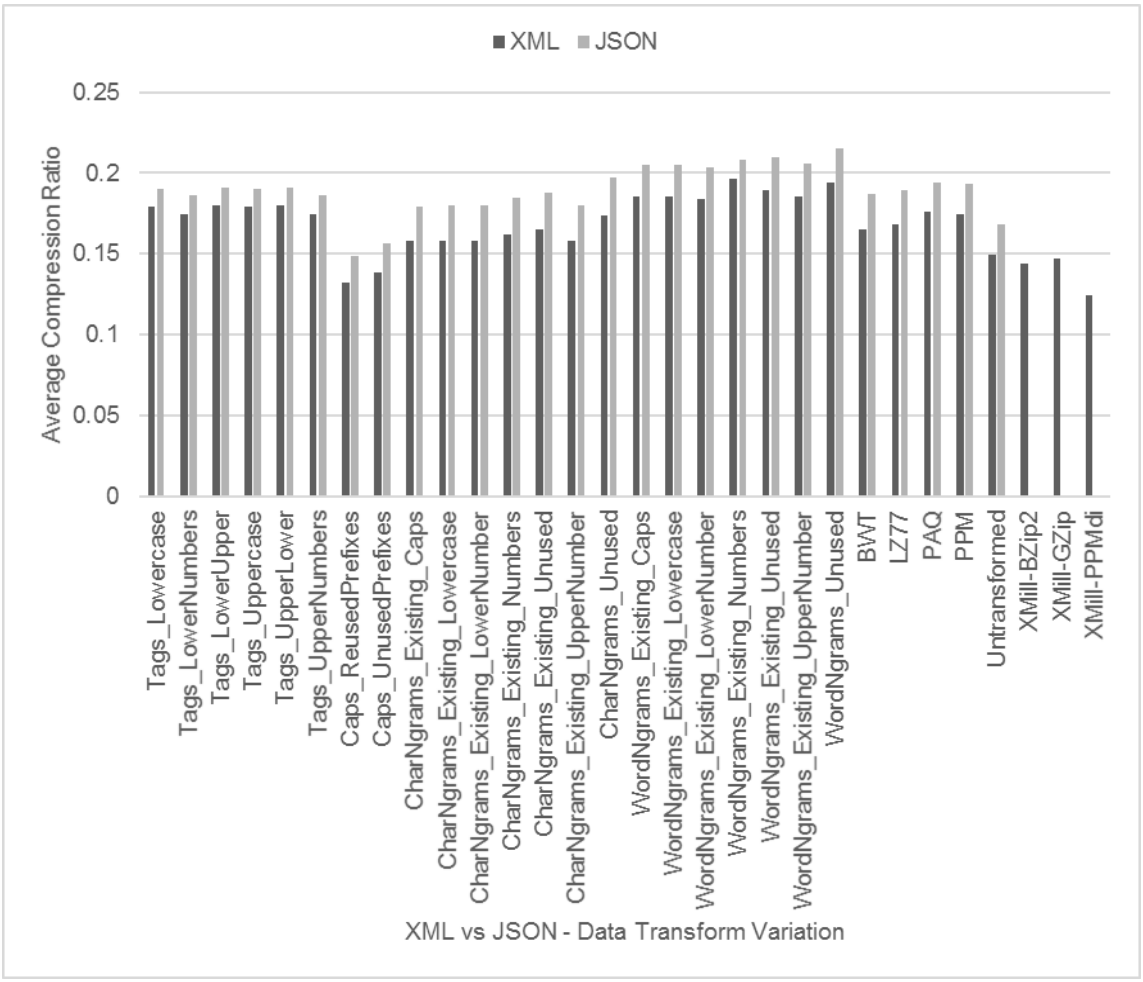

Figure 3. Overall Average Compression Ratios for XML AND JSON Data Transform Variations

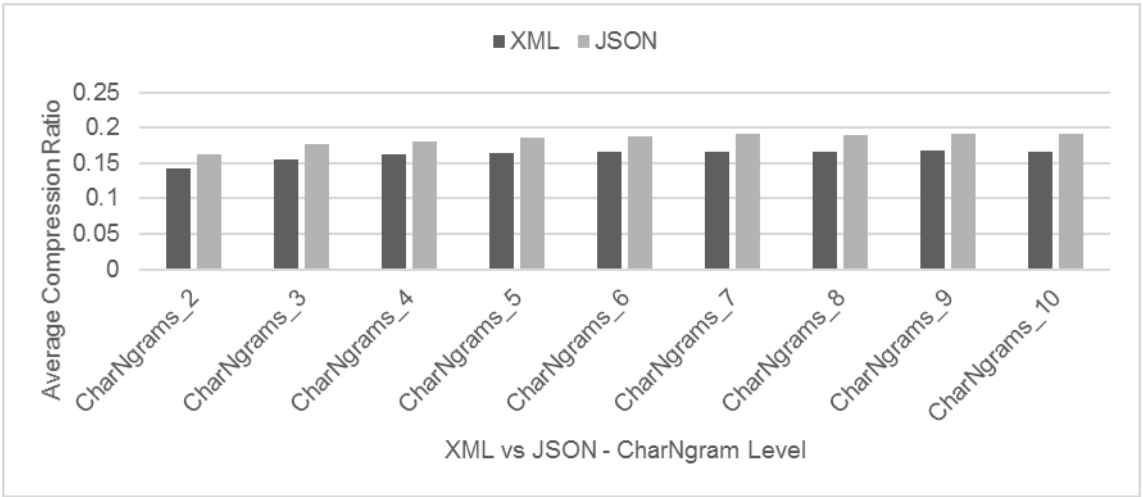

Figure 4. Overall Average Compression Ratios for XML AND JSON CharNgram Levels

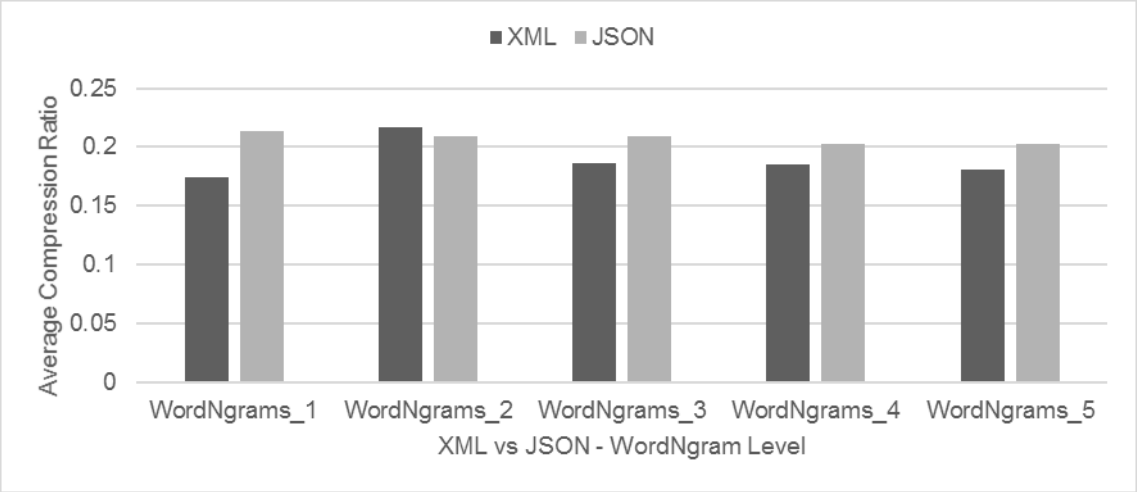

Figure 5. Overall Average Compression Ratios for XML AND JSON WordNgram Levels

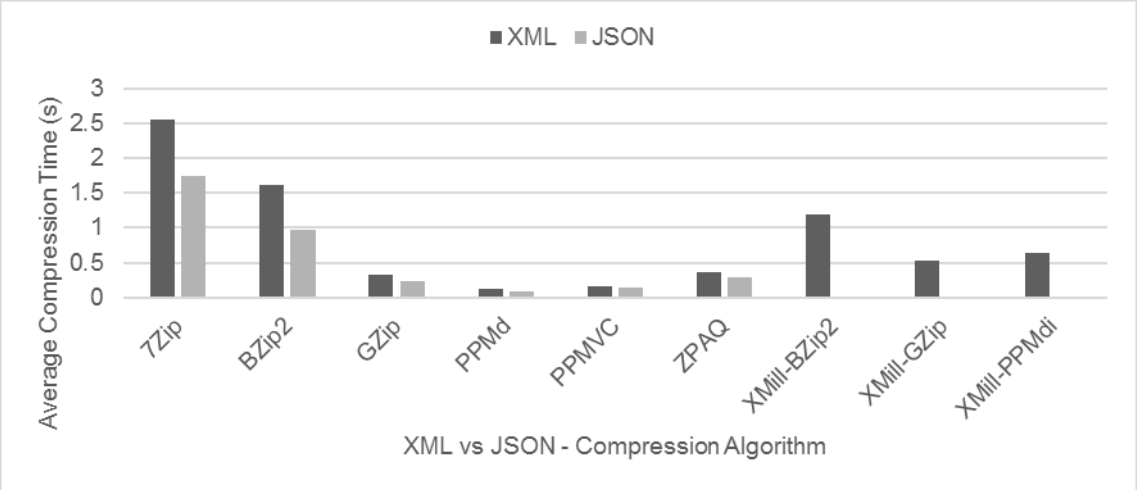

Figure 6. Overall Average Compression Times (s) for XML AND JSON Compression Algorithms

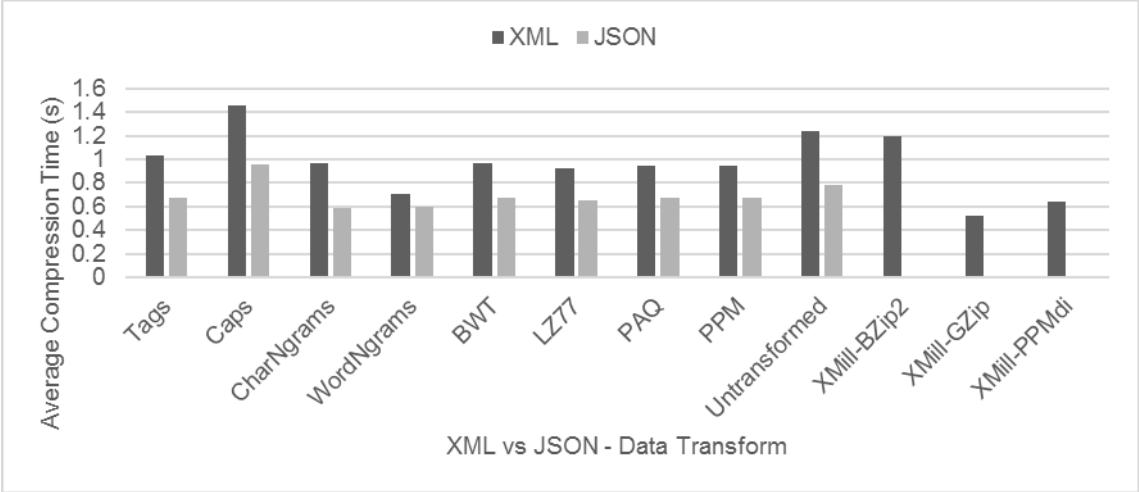

Figure 7. Overall Average Compression Times (s) for XML AND JSON Data Transforms

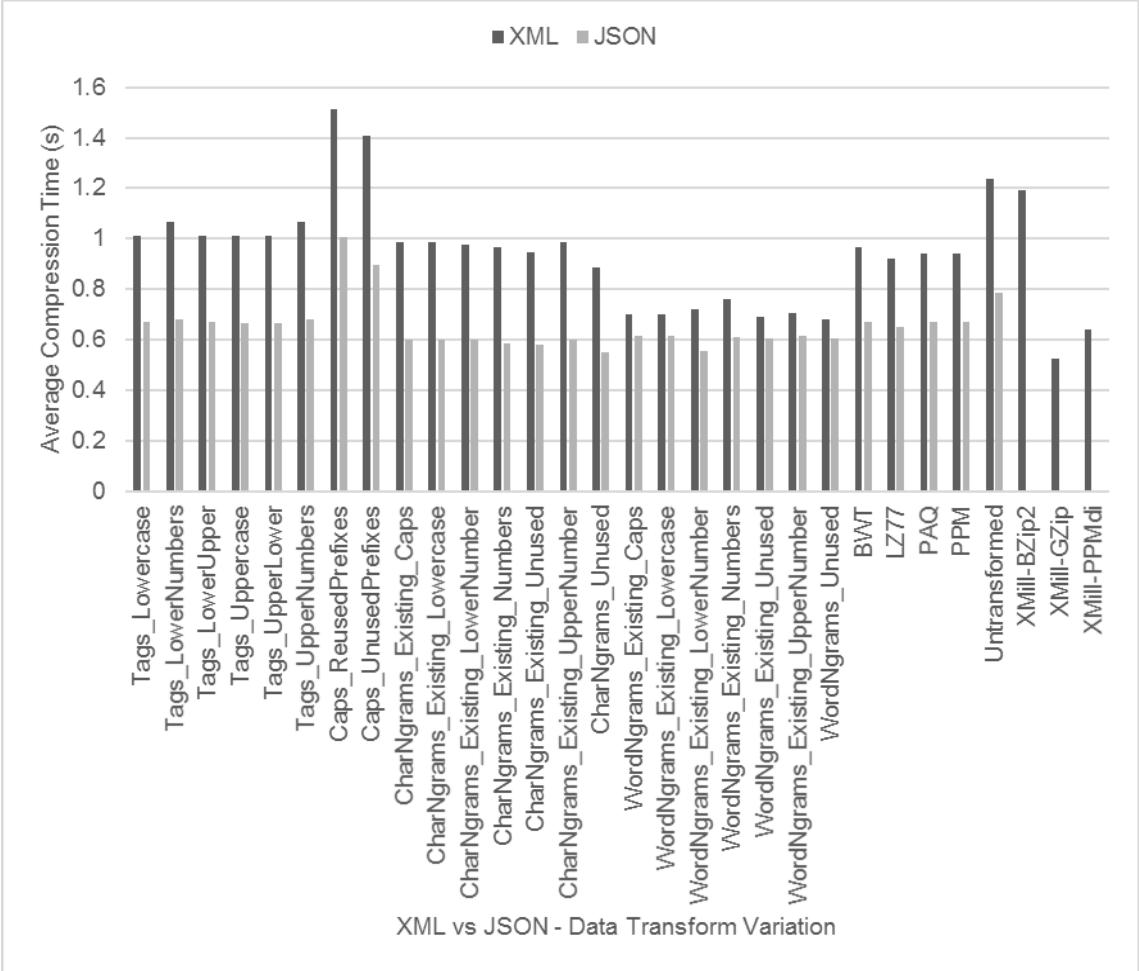

Figure 8. Overall Average Compression Times (s) for XML AND JSON Data Transform Variations

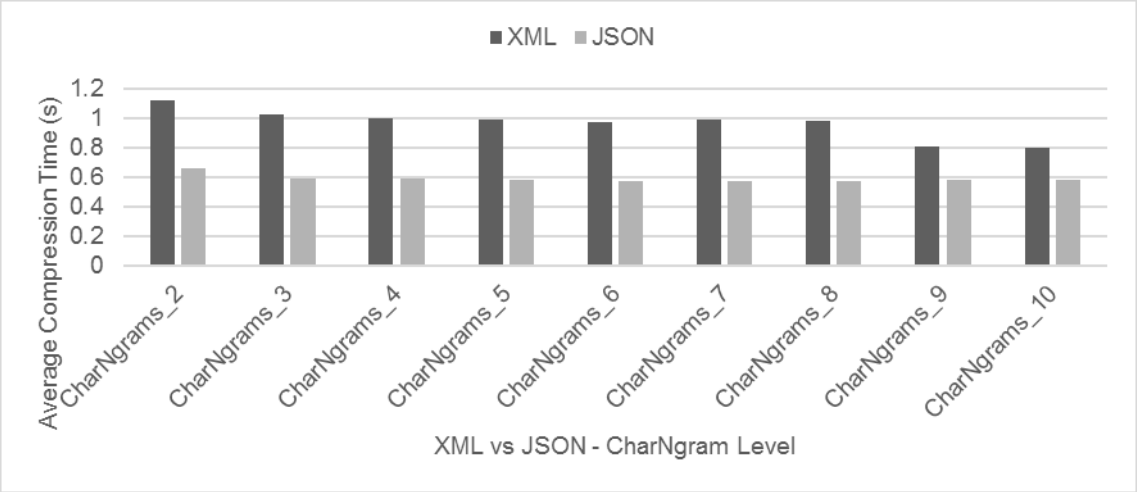

Figure 9. Overall Average Compression Times (s) for XML AND JSON CharNgram Levels

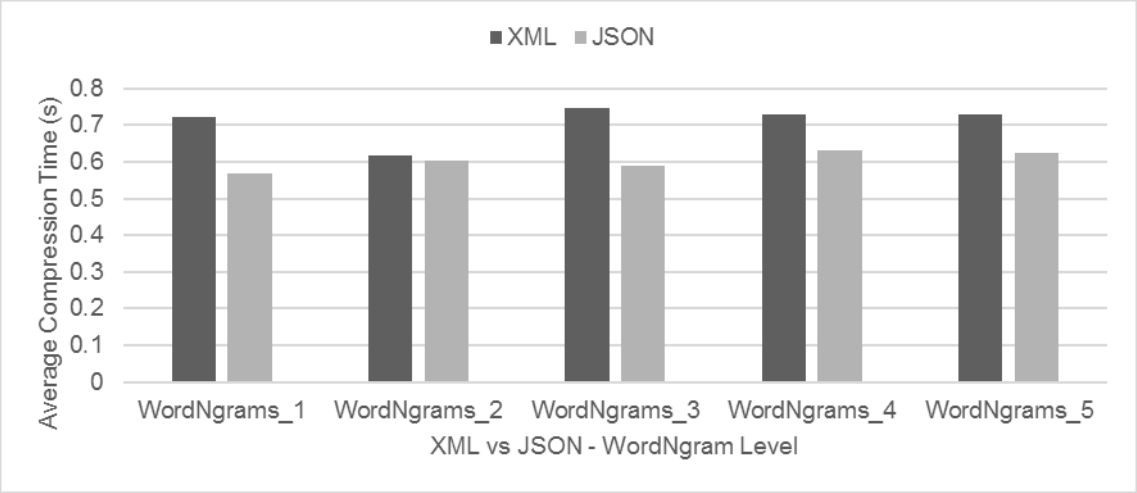

Figure 10. Overall Average Compression Times (s) for XML AND JSON WordNgram Levels

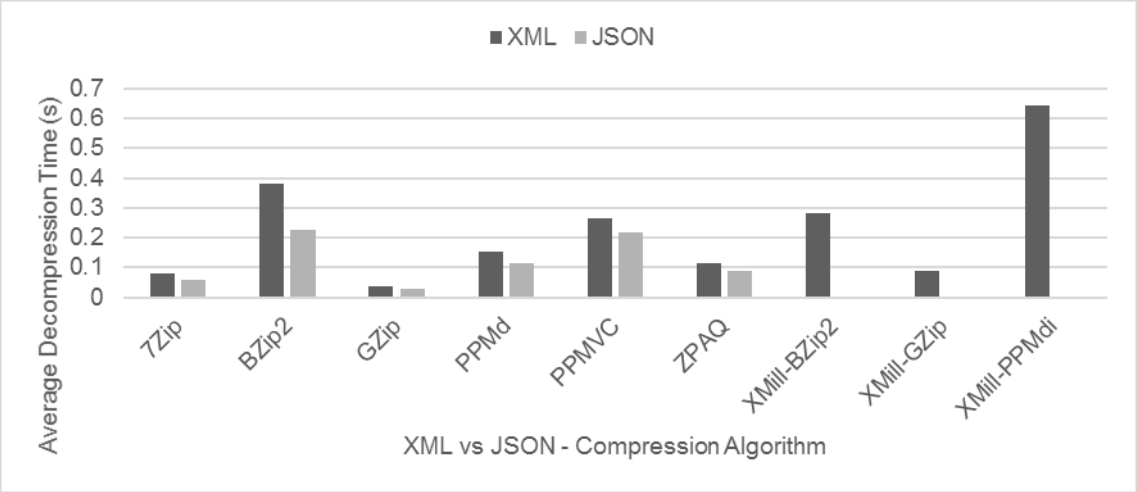

Figure 11. Overall Average Decompression Times (s) for XML AND JSON Compression Algorithms

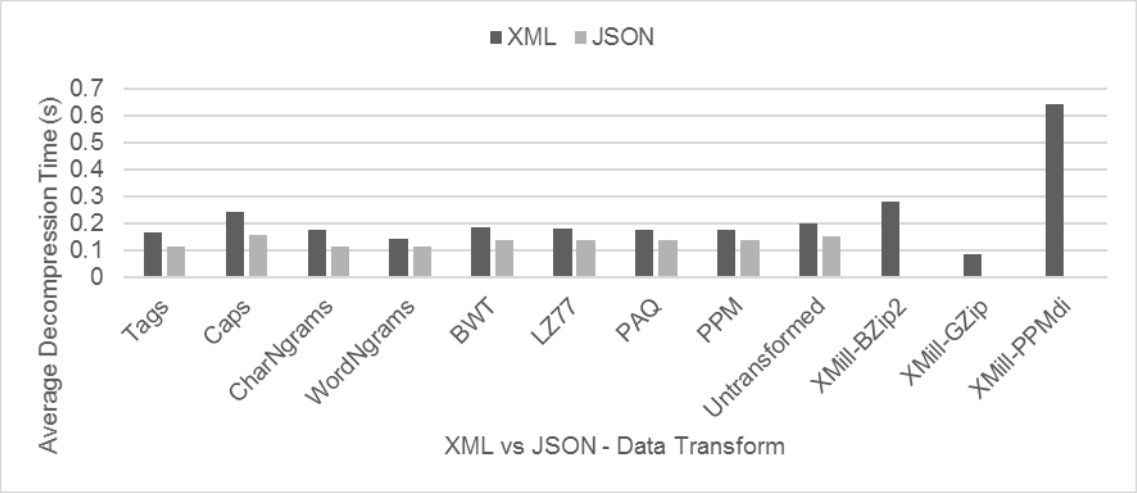

Figure 12. Overall Average Decompression Times (s) for XML AND JSON Data Transforms

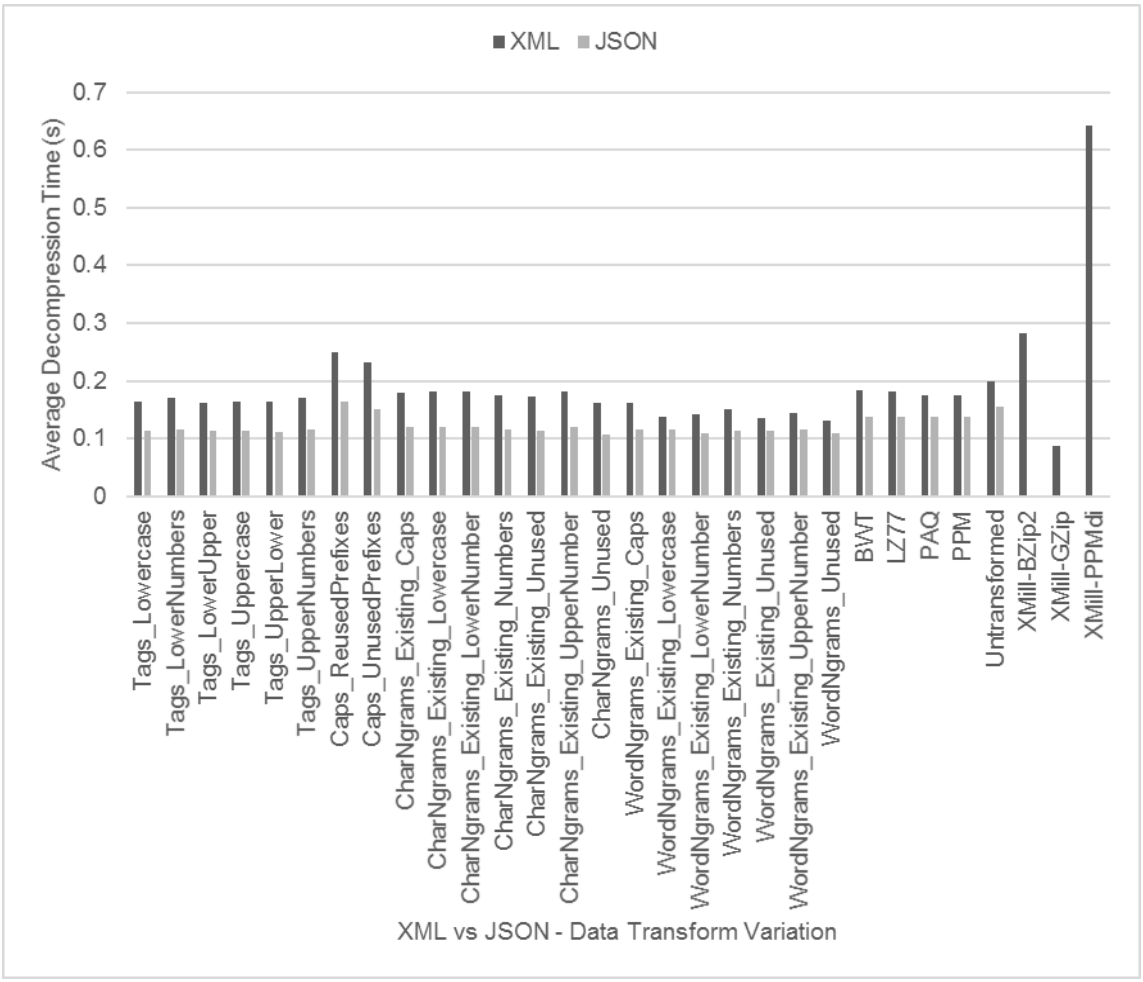

Figure 13. Overall Average Decompression Times (s) for XML AND JSON Data Transform Variations

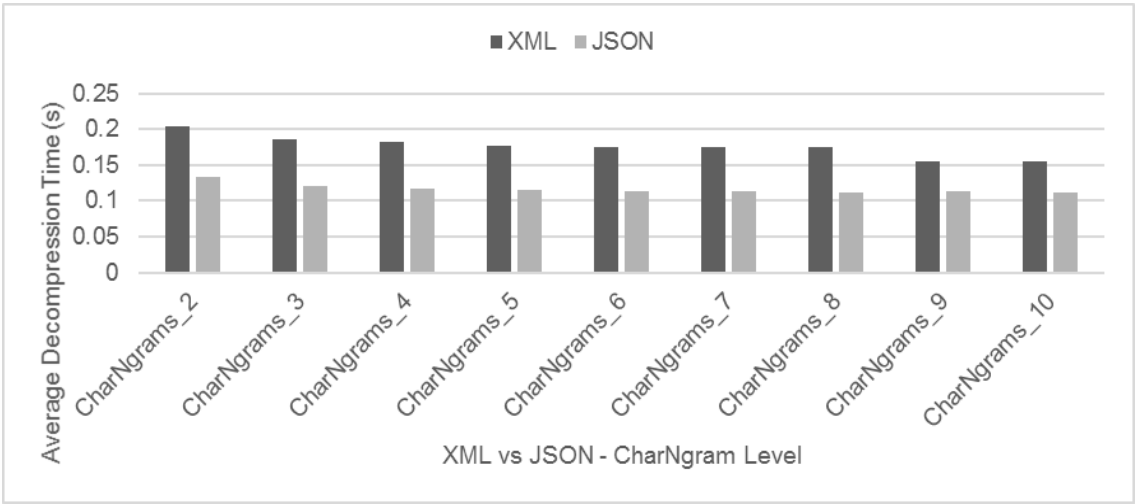

Figure 14. Overall Average Decompression Times (s) for XML AND JSON CharNgram Levels

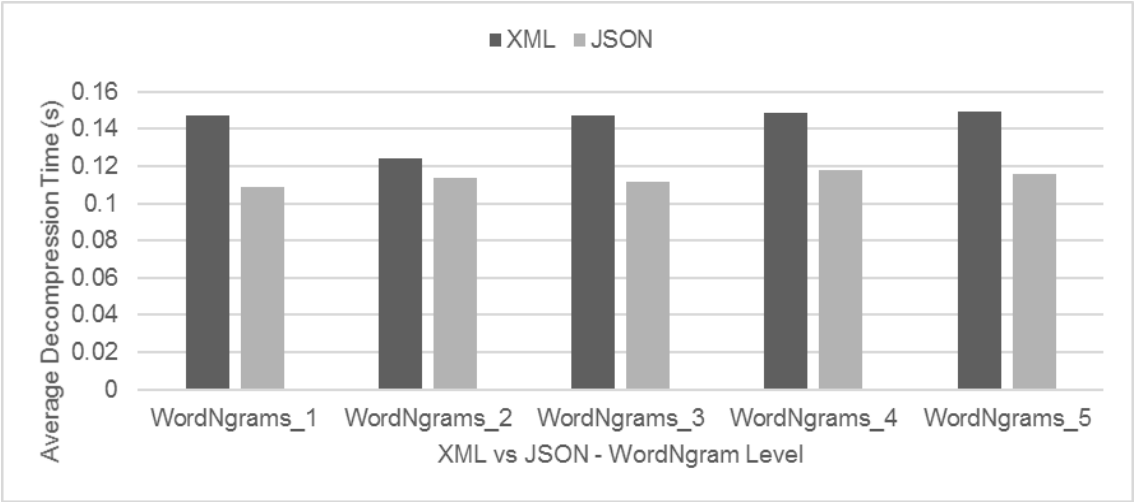

Figure 15. Overall Average Decompression Times (s) for XML AND JSON WordNgram Levels

Table 1. Average Compression Ratio for XML Data Transform

	XML - Average Compression Ratio						
Data Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
Tags	0.161	0.151	0.184	0.163	0.148	0.250	
Caps	0.124	0.115	0.139	0.124	0.117	0.187	
CharNgrams	0.149	0.139	0.166	0.145	0.140	0.223	
WordNgrams	0.176	0.162	0.193	0.163	0.157	0.269	
BWT	0.153	0.146	0.171	0.148	0.148	0.221	
LZ77	0.155	0.150	0.172	0.151	0.151	0.223	
PAQ	0.161	0.152	0.184	0.158	0.159	0.237	
PPM	0.162	0.151	0.182	0.155	0.156	0.236	
Untransformed	0.137	0.129	0.154	0.133	0.128	0.205	
XMill-BZip2							0.144
XMill-GZip							0.147
XMill-PPMdi							0.124
Total Average	0.158	0.147	0.175	0.152	0.146	0.238	0.144

Table 2. Average Compression Ratio for XML Data Transform Variation

	XML - Average Compression Ratio						
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
Tags_Lowercase	0.163	0.153	0.186	0.163	0.149	0.253	
Tags_LowerNumbers	0.157	0.147	0.180	0.163	0.145	0.244	
Tags_LowerUpper	0.163	0.153	0.186	0.163	0.150	0.253	
Tags_Uppercase	0.163	0.153	0.186	0.163	0.149	0.253	
Tags_UpperLower	0.163	0.153	0.186	0.163	0.150	0.252	
Tags_UpperNumbers	0.157	0.147	0.180	0.163	0.145	0.244	
Caps_ReusedPrefixes	0.120	0.111	0.135	0.125	0.114	0.181	
Caps_UnusedPrefixes	0.127	0.118	0.142	0.124	0.120	0.192	
CharNgrams_Existing_Caps	0.146	0.135	0.162	0.142	0.136	0.218	
CharNgrams_Existing_Lowercase	0.146	0.136	0.162	0.142	0.137	0.218	
CharNgrams_Existing_LowerNumber	0.146	0.135	0.162	0.142	0.137	0.218	
CharNgrams_Existing_Numbers	0.150	0.139	0.166	0.146	0.140	0.224	
CharNgrams_Existing_Unused	0.153	0.142	0.170	0.148	0.143	0.228	
CharNgrams_Existing_UpperNumber	0.146	0.135	0.162	0.142	0.137	0.218	
CharNgrams_Unused	0.160	0.150	0.178	0.155	0.149	0.239	
WordNgrams_Existing_Caps	0.172	0.158	0.190	0.160	0.156	0.265	
WordNgrams_Existing_Lowercase	0.173	0.158	0.190	0.161	0.156	0.265	
WordNgrams_Existing_LowerNumber	0.171	0.156	0.188	0.159	0.155	0.262	
WordNgrams_Existing_Numbers	0.185	0.172	0.203	0.171	0.156	0.282	

Table 2. Average Compression Ratio for XML Data Transform Variation

	XML - Average Compression Ratio						
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
WordNgrams_Existing_Unused	0.177	0.163	0.195	0.163	0.159	0.271	
WordNgrams_Existing_UpperNumber	0.173	0.158	0.190	0.160	0.156	0.264	
WordNgrams_Unused	0.181	0.168	0.198	0.167	0.163	0.278	
BWT	0.153	0.146	0.171	0.148	0.148	0.221	
LZ77	0.155	0.150	0.172	0.151	0.151	0.223	
PAQ	0.161	0.152	0.184	0.158	0.159	0.237	
PPM	0.162	0.151	0.182	0.155	0.156	0.236	
Untransformed	0.137	0.129	0.154	0.133	0.128	0.205	
XMill-BZip2							0.144
XMill-GZip							0.147
XMill-PPMdi							0.124
Total Average	0.158	0.147	0.175	0.152	0.146	0.238	0.144

Table 3. Average Compression Ratio for XML CharNgram Level

	XML - Average Compression Ratio					
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	0.133	0.121	0.148	0.129	0.123	0.195
CharNgrams_3	0.144	0.134	0.160	0.138	0.135	0.214
CharNgrams_4	0.149	0.140	0.166	0.145	0.140	0.223
CharNgrams_5	0.151	0.142	0.168	0.147	0.142	0.226
CharNgrams_6	0.152	0.143	0.170	0.148	0.143	0.228
CharNgrams_7	0.152	0.142	0.169	0.148	0.142	0.228
CharNgrams_8	0.154	0.144	0.171	0.150	0.144	0.231
CharNgrams_9	0.155	0.144	0.171	0.151	0.145	0.232
CharNgrams_10	0.154	0.143	0.171	0.151	0.144	0.231
Total Average	0.149	0.139	0.166	0.145	0.140	0.223

Table 4. Average Compression Ratio for XML WordNgram Level

	XML - Average Compression Ratio					
WordNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_1	0.162	0.147	0.179	0.157	0.144	0.246
WordNgrams_2	0.202	0.188	0.224	0.183	0.177	0.311
WordNgrams_3	0.173	0.161	0.191	0.160	0.159	0.263
WordNgrams_4	0.173	0.159	0.189	0.159	0.156	0.266
WordNgrams_5	0.169	0.154	0.184	0.155	0.151	0.260
Total Average	0.176	0.162	0.193	0.163	0.157	0.269

Table 5. Average Compression Time (s) for XML Data Transform

	XML - Average Compression Times (s)						
Data Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
Tags	3.354	1.474	0.404	0.102	0.128	0.387	
Caps	4.266	2.682	0.498	0.158	0.179	0.487	
CharNgrams	2.671	1.810	0.330	0.127	0.159	0.382	
WordNgrams	2.051	1.168	0.262	0.104	0.148	0.318	
BWT	2.375	2.082	0.339	0.143	0.155	0.413	
LZ77	2.244	1.967	0.328	0.141	0.152	0.401	
PAQ	2.558	1.755	0.362	0.128	0.152	0.397	
PPM	2.572	1.738	0.365	0.128	0.153	0.400	
Untransformed	3.634	2.203	0.429	0.136	0.173	0.449	
XMill-BZip2							1.190
XMill-GZip							0.524
XMill-PPMdi							0.642
Total Average	2.545	1.615	0.318	0.120	0.154	0.366	1.190

Table 6. Average Compression Time (s) for XML Data Transform Variation

	XML - Average Compression Times (s)						
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
Tags_Lowercase	3.329	1.394	0.403	0.101	0.130	0.383	
Tags_LowerNumbers	3.403	1.614	0.406	0.107	0.132	0.394	
Tags_LowerUpper	3.330	1.405	0.402	0.099	0.126	0.384	
Tags_Uppercase	3.328	1.402	0.402	0.098	0.126	0.384	
Tags_UpperLower	3.331	1.403	0.404	0.098	0.127	0.385	
Tags_UpperNumbers	3.403	1.627	0.406	0.107	0.130	0.394	
Caps_ReusedPrefixes	4.404	2.783	0.533	0.167	0.184	0.497	
Caps_UnusedPrefixes	4.129	2.581	0.463	0.150	0.174	0.477	
CharNgrams_Existing_Caps	2.719	1.872	0.341	0.132	0.161	0.388	
CharNgrams_Existing_Lowercase	2.726	1.874	0.341	0.132	0.161	0.388	
CharNgrams_Existing_LowerNumber	2.723	1.832	0.337	0.132	0.160	0.388	
CharNgrams_Existing_Numbers	2.676	1.843	0.328	0.130	0.159	0.381	
CharNgrams_Existing_Unused	2.626	1.777	0.322	0.126	0.157	0.378	
CharNgrams_Existing_UpperNumber	2.729	1.879	0.337	0.132	0.160	0.389	
CharNgrams_Unused	2.494	1.593	0.305	0.108	0.154	0.365	
WordNgrams_Existing_Caps	2.021	1.164	0.259	0.104	0.148	0.315	
WordNgrams_Existing_Lowercase	2.029	1.161	0.259	0.104	0.148	0.315	
WordNgrams_Existing_LowerNumber	2.094	1.185	0.261	0.106	0.151	0.324	
WordNgrams_Existing_Numbers	2.270	1.247	0.291	0.115	0.148	0.338	
WordNgrams_Existing_Unused	2.000	1.135	0.256	0.102	0.147	0.313	
WordNgrams_Existing_UpperNumber	2.024	1.173	0.261	0.105	0.148	0.316	

Table 6. Average Compression Time (s) for XML Data Transform Variation

	XML - Average Compression Times (s)						
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
WordNgrams_Unused	1.943	1.121	0.250	0.091	0.146	0.308	
BWT	2.375	2.082	0.339	0.143	0.155	0.413	
LZ77	2.244	1.967	0.328	0.141	0.152	0.401	
PAQ	2.558	1.755	0.362	0.128	0.152	0.397	
PPM	2.572	1.738	0.365	0.128	0.153	0.400	
Untransformed	3.634	2.203	0.429	0.136	0.173	0.449	
XMill-BZip2							1.190
XMill-GZip							0.524
XMill-PPMdi							0.642
Total Average	2.545	1.615	0.318	0.120	0.154	0.366	1.190

Table 7. Average Compression Time (s) for XML CharNgram Level

	XML - Average Compression Times (s)					
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	2.992	2.262	0.386	0.147	0.169	0.427
CharNgrams_3	2.749	2.026	0.330	0.133	0.163	0.400
CharNgrams_4	2.778	1.886	0.337	0.129	0.159	0.391
CharNgrams_5	2.789	1.807	0.340	0.124	0.158	0.387
CharNgrams_6	2.810	1.727	0.345	0.123	0.157	0.384
CharNgrams_7	2.837	1.749	0.348	0.123	0.156	0.385
CharNgrams_8	2.834	1.708	0.349	0.122	0.156	0.384
CharNgrams_9	2.123	1.569	0.267	0.123	0.156	0.342
CharNgrams_10	2.122	1.556	0.267	0.124	0.157	0.341
Total Average	2.671	1.810	0.330	0.127	0.159	0.382

Table 8. Average Compression Time (s) for XML WordNgram Level

	XML - Average Compression Times (s)					
WordNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_1	2.097	1.167	0.271	0.107	0.152	0.327
WordNgrams_2	1.916	0.845	0.256	0.087	0.136	0.293
WordNgrams_3	2.080	1.334	0.260	0.112	0.149	0.326
WordNgrams_4	2.081	1.254	0.262	0.107	0.151	0.322
WordNgrams_5	2.080	1.240	0.261	0.107	0.152	0.322
Total Average	2.051	1.168	0.262	0.104	0.148	0.318

Table 9. Average Decompression Time (s) for XML Data Transform

	XML - Average Decompression Times (s)						
Data Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
Tags	0.098	0.408	0.041	0.128	0.215	0.121	
Caps	0.118	0.595	0.056	0.209	0.343	0.156	
CharNgrams	0.086	0.401	0.040	0.164	0.280	0.120	
WordNgrams	0.067	0.320	0.029	0.132	0.246	0.097	
BWT	0.113	0.408	0.038	0.182	0.269	0.124	
LZ77	0.113	0.395	0.045	0.179	0.264	0.127	
PAQ	0.110	0.384	0.049	0.162	0.257	0.119	
PPM	0.109	0.384	0.045	0.162	0.260	0.124	
Untransformed	0.111	0.473	0.028	0.176	0.290	0.147	
XMill-BZip2							0.282
XMill-GZip							0.088
XMill-PPMdi							0.642
Total Average	0.082	0.380	0.037	0.154	0.266	0.114	0.282

Table 10. Average Decompression Time (s) for XML Data Transform Variation

	XML - Average Decompression Times (s)						
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
Tags_Lowercase	0.097	0.406	0.039	0.125	0.212	0.120	
Tags_LowerNumbers	0.100	0.420	0.046	0.135	0.221	0.124	
Tags_LowerUpper	0.097	0.399	0.039	0.125	0.211	0.120	
Tags_Uppercase	0.097	0.401	0.042	0.124	0.211	0.121	
Tags_UpperLower	0.097	0.404	0.039	0.124	0.211	0.120	
Tags_UpperNumbers	0.098	0.419	0.041	0.135	0.221	0.120	
Caps_ReusedPrefixes	0.121	0.620	0.059	0.221	0.359	0.159	
Caps_UnusedPrefixes	0.116	0.570	0.054	0.197	0.327	0.152	
CharNgrams_Existing_Caps	0.087	0.411	0.039	0.170	0.285	0.121	
CharNgrams_Existing_Lowercase	0.087	0.414	0.041	0.170	0.286	0.123	
CharNgrams_Existing_LowerNumber	0.087	0.408	0.040	0.170	0.285	0.127	
CharNgrams_Existing_Numbers	0.085	0.394	0.040	0.165	0.279	0.121	
CharNgrams_Existing_Unused	0.085	0.391	0.038	0.162	0.274	0.116	
CharNgrams_Existing_UpperNumber	0.086	0.418	0.042	0.171	0.284	0.122	
CharNgrams_Unused	0.084	0.371	0.037	0.139	0.263	0.111	
WordNgrams_Existing_Caps	0.066	0.422	0.030	0.133	0.246	0.095	
WordNgrams_Existing_Lowercase	0.066	0.290	0.028	0.133	0.247	0.095	
WordNgrams_Existing_LowerNumber	0.067	0.298	0.031	0.135	0.253	0.104	
WordNgrams_Existing_Numbers	0.073	0.327	0.030	0.146	0.245	0.102	
WordNgrams_Existing_Unused	0.065	0.288	0.028	0.130	0.247	0.094	
WordNgrams_Existing_UpperNumber	0.066	0.328	0.028	0.133	0.247	0.096	

Table 10. Average Decompression Time (s) for XML Data Transform Variation

	XML - Average Decompression Times (s)						
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ	XMill
WordNgrams_Unused	0.065	0.287	0.026	0.115	0.240	0.093	
BWT	0.113	0.408	0.038	0.182	0.269	0.124	
LZ77	0.113	0.395	0.045	0.179	0.264	0.127	
PAQ	0.110	0.384	0.049	0.162	0.257	0.119	
PPM	0.109	0.384	0.045	0.162	0.260	0.124	
Untransformed	0.111	0.473	0.028	0.176	0.290	0.147	
XMill-BZip2							0.282
XMill-GZip							0.088
XMill-PPMdi							0.642
Total Average	0.082	0.380	0.037	0.154	0.266	0.114	0.282

Table 11. Average Decompression Time (s) for XML CharNgram Level

	XML - Average Decompression Times (s)					
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	0.096	0.479	0.046	0.193	0.312	0.136
CharNgrams_3	0.092	0.429	0.042	0.172	0.291	0.127
CharNgrams_4	0.090	0.416	0.041	0.165	0.281	0.127
CharNgrams_5	0.089	0.408	0.041	0.160	0.278	0.121
CharNgrams_6	0.089	0.403	0.042	0.157	0.273	0.120
CharNgrams_7	0.089	0.406	0.040	0.158	0.270	0.119
CharNgrams_8	0.089	0.404	0.040	0.157	0.270	0.121
CharNgrams_9	0.069	0.334	0.034	0.159	0.270	0.105
CharNgrams_10	0.069	0.331	0.031	0.158	0.271	0.105
Total Average	0.086	0.401	0.040	0.164	0.280	0.120

Table 12. Average Decompression Time (s) for XML WordNgram Level

	XML - Average Decompression Times (s)					
WordNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_1	0.068	0.330	0.025	0.137	0.256	0.103
WordNgrams_2	0.063	0.268	0.028	0.109	0.217	0.088
WordNgrams_3	0.067	0.327	0.029	0.143	0.249	0.099
WordNgrams_4	0.067	0.335	0.032	0.136	0.253	0.098
WordNgrams_5	0.068	0.341	0.030	0.136	0.257	0.098
Total Average	0.067	0.320	0.029	0.132	0.246	0.097

Table 13. Average Compression Ratio for JSON Data Transform

	JSON - Average Compression Ratio					
Data Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Tags	0.173	0.161	0.197	0.167	0.152	0.272
Caps	0.141	0.129	0.158	0.134	0.127	0.216
CharNgrams	0.172	0.158	0.191	0.157	0.152	0.262
WordNgrams	0.194	0.177	0.214	0.173	0.171	0.303
BWT	0.175	0.164	0.196	0.162	0.159	0.257
LZ77	0.176	0.169	0.197	0.165	0.162	0.259
PAQ	0.179	0.168	0.204	0.169	0.169	0.269
PPM	0.181	0.167	0.203	0.166	0.166	0.268
Untransformed	0.156	0.145	0.175	0.145	0.140	0.239
Average Total	0.178	0.164	0.198	0.163	0.158	0.274

Table 14. Average Compression Ratio for JSON Data Transform Variation

	JSON - Average Compression Ratio					
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Tags_Lowercase	0.174	0.163	0.198	0.168	0.153	0.274
Tags_LowerNumbers	0.171	0.158	0.194	0.165	0.150	0.268
Tags_LowerUpper	0.175	0.163	0.198	0.168	0.153	0.274
Tags_Uppercase	0.175	0.163	0.199	0.167	0.153	0.274
Tags_UpperLower	0.175	0.163	0.198	0.167	0.153	0.275
Tags_UpperNumbers	0.171	0.158	0.194	0.165	0.150	0.268
Caps_ReusedPrefixes	0.137	0.125	0.154	0.134	0.124	0.209
Caps_UnusedPrefixes	0.144	0.133	0.162	0.135	0.130	0.223
CharNgrams_Existing_Caps	0.167	0.154	0.187	0.154	0.149	0.256
CharNgrams_Existing_Lowercase	0.168	0.154	0.187	0.154	0.149	0.256
CharNgrams_Existing_LowerNumber	0.168	0.154	0.187	0.154	0.149	0.256
CharNgrams_Existing_Numbers	0.172	0.159	0.192	0.158	0.153	0.263
CharNgrams_Existing_Unused	0.176	0.162	0.196	0.160	0.156	0.268
CharNgrams_Existing_UpperNumber	0.168	0.154	0.187	0.154	0.149	0.256
CharNgrams_Unused	0.184	0.172	0.205	0.167	0.163	0.281
WordNgrams_Existing_Caps	0.191	0.175	0.212	0.172	0.169	0.299
WordNgrams_Existing_Lowercase	0.192	0.175	0.212	0.172	0.169	0.299
WordNgrams_Existing_LowerNumber	0.191	0.173	0.210	0.171	0.168	0.297
WordNgrams_Existing_Numbers	0.194	0.178	0.214	0.174	0.171	0.303
WordNgrams_Existing_Unused	0.196	0.180	0.217	0.175	0.173	0.306

Table 14. Average Compression Ratio for JSON Data Transform Variation

	JSON - Average Compression Ratio					
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_Existing_UpperNumber	0.192	0.175	0.212	0.172	0.169	0.299
WordNgrams_Unused	0.200	0.185	0.221	0.179	0.176	0.314
BWT	0.175	0.164	0.196	0.162	0.159	0.257
LZ77	0.176	0.169	0.197	0.165	0.162	0.259
PAQ	0.179	0.168	0.204	0.169	0.169	0.269
PPM	0.181	0.167	0.203	0.166	0.166	0.268
Untransformed	0.156	0.145	0.175	0.145	0.140	0.239
Average Total	0.178	0.164	0.198	0.163	0.158	0.274

Table 15. Average Compression Ratio for JSON CharNgram Level

	JSON - Average Compression Ratio					
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	0.152	0.137	0.169	0.140	0.134	0.227
CharNgrams_3	0.165	0.152	0.184	0.149	0.147	0.250
CharNgrams_4	0.169	0.157	0.188	0.154	0.150	0.257
CharNgrams_5	0.173	0.160	0.193	0.158	0.154	0.264
CharNgrams_6	0.176	0.163	0.196	0.161	0.156	0.269
CharNgrams_7	0.178	0.165	0.198	0.163	0.158	0.273
CharNgrams_8	0.177	0.163	0.197	0.162	0.156	0.271
CharNgrams_9	0.178	0.164	0.198	0.164	0.158	0.274
CharNgrams_10	0.178	0.165	0.198	0.164	0.159	0.273
Average Total	0.172	0.158	0.191	0.157	0.152	0.262

Table 16. Average Compression Ratio for JSON WordNgram Level

	JSON - Average Compression Ratio					
WordNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_1	0.200	0.182	0.221	0.181	0.173	0.312
WordNgrams_2	0.196	0.180	0.217	0.174	0.176	0.302
WordNgrams_3	0.195	0.179	0.216	0.174	0.172	0.304
WordNgrams_4	0.190	0.173	0.209	0.169	0.168	0.296
WordNgrams_5	0.189	0.172	0.208	0.169	0.164	0.298
Average Total	0.194	0.177	0.214	0.173	0.171	0.303

Table 17. Average Compression Time (s) for JSON Data Transform

	JSON - Average Compression Times (s)					
Data Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Tags	2.087	0.988	0.259	0.086	0.118	0.300
Caps	2.687	1.762	0.326	0.119	0.162	0.364
CharNgrams	1.634	0.987	0.217	0.096	0.147	0.291
WordNgrams	1.836	0.851	0.242	0.088	0.141	0.291
BWT	1.741	1.238	0.247	0.107	0.198	0.318
LZ77	1.698	1.197	0.243	0.106	0.196	0.313
PAQ	1.810	1.180	0.254	0.102	0.198	0.313
PPM	1.812	1.174	0.258	0.103	0.199	0.314
Untransformed	2.193	1.403	0.278	0.104	0.212	0.329
Average Total	1.751	0.970	0.231	0.094	0.146	0.294

Table 18. Average Compression Time (s) for JSON Data Transform Variation

	JSON - Average Compression Times (s)					
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Tags_Lowercase	2.082	0.974	0.259	0.085	0.118	0.299
Tags_LowerNumbers	2.106	1.015	0.260	0.088	0.118	0.303
Tags_LowerUpper	2.079	0.976	0.258	0.085	0.118	0.299
Tags_Uppercase	2.075	0.972	0.259	0.085	0.117	0.298
Tags_UpperLower	2.074	0.974	0.259	0.085	0.118	0.298
Tags_UpperNumbers	2.107	1.016	0.260	0.088	0.118	0.303
Caps_ReusedPrefixes	2.788	1.926	0.351	0.125	0.165	0.372
Caps_UnusedPrefixes	2.586	1.598	0.301	0.114	0.159	0.355
CharNgrams_Existing_Caps	1.662	1.025	0.225	0.100	0.149	0.294
CharNgrams_Existing_Lowercase	1.665	1.023	0.224	0.100	0.149	0.294
CharNgrams_Existing_LowerNumber	1.668	1.011	0.221	0.100	0.148	0.295
CharNgrams_Existing_Numbers	1.635	0.974	0.218	0.098	0.147	0.291
CharNgrams_Existing_Unused	1.603	0.975	0.212	0.095	0.146	0.287
CharNgrams_Existing_UpperNumber	1.668	1.020	0.221	0.100	0.148	0.295
CharNgrams_Unused	1.537	0.882	0.202	0.081	0.143	0.280
WordNgrams_Existing_Caps	1.875	0.865	0.249	0.089	0.142	0.294
WordNgrams_Existing_Lowercase	1.880	0.872	0.248	0.089	0.142	0.294
WordNgrams_Existing_LowerNumber	1.652	0.807	0.219	0.091	0.146	0.280
WordNgrams_Existing_Numbers	1.869	0.869	0.246	0.089	0.142	0.293
WordNgrams_Existing_Unused	1.858	0.844	0.245	0.088	0.140	0.291

Table 18. Average Compression Time (s) for JSON Data Transform Variation

	JSON - Average Compression Times (s)					
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_Existing_UpperNumber	1.877	0.863	0.248	0.089	0.142	0.294
WordNgrams_Unused	1.838	0.838	0.243	0.077	0.138	0.289
BWT	1.741	1.238	0.247	0.107	0.198	0.318
LZ77	1.698	1.197	0.243	0.106	0.196	0.313
PAQ	1.810	1.180	0.254	0.102	0.198	0.313
PPM	1.812	1.174	0.258	0.103	0.199	0.314
Untransformed	2.193	1.403	0.278	0.104	0.212	0.329
Average Total	1.751	0.970	0.231	0.094	0.146	0.294

Table 19. Average Compression Time (s) for JSON CharNgram Level

	JSON - Average Compression Times (s)					
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	1.740	1.219	0.236	0.112	0.157	0.314
CharNgrams_3	1.566	1.067	0.202	0.100	0.149	0.296
CharNgrams_4	1.591	1.047	0.209	0.098	0.148	0.293
CharNgrams_5	1.605	0.990	0.213	0.095	0.146	0.289
CharNgrams_6	1.622	0.930	0.217	0.093	0.145	0.287
CharNgrams_7	1.638	0.903	0.219	0.092	0.145	0.285
CharNgrams_8	1.638	0.902	0.220	0.092	0.144	0.284
CharNgrams_9	1.655	0.918	0.220	0.093	0.144	0.285
CharNgrams_10	1.651	0.909	0.221	0.093	0.146	0.285
Average Total	1.634	0.987	0.217	0.096	0.147	0.291

Table 20. Average Compression Time (s) for JSON WordNgram Level

	JSON - Average Compression Times (s)					
WordNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_1	1.792	0.704	0.247	0.083	0.141	0.283
WordNgrams_2	1.832	0.863	0.240	0.089	0.141	0.291
WordNgrams_3	1.836	0.790	0.239	0.084	0.140	0.286
WordNgrams_4	1.861	0.968	0.243	0.092	0.144	0.298
WordNgrams_5	1.859	0.931	0.242	0.090	0.142	0.296
Average Total	1.836	0.851	0.242	0.088	0.141	0.291

Table 21. Average Decompression Time (s) for JSON Data Transform

JSON - Average Decompression Times (s)						
Data Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Tags	0.064	0.237	0.029	0.107	0.183	0.088
Caps	0.077	0.337	0.036	0.153	0.261	0.114
CharNgrams	0.057	0.220	0.028	0.119	0.221	0.090
WordNgrams	0.060	0.223	0.027	0.107	0.202	0.090
BWT	0.071	0.262	0.024	0.133	0.300	0.094
LZ77	0.070	0.257	0.027	0.132	0.297	0.093
PAQ	0.070	0.258	0.030	0.127	0.300	0.093
PPM	0.070	0.262	0.027	0.128	0.300	0.093
Untransformed	0.074	0.324	0.016	0.136	0.340	0.099
Average Total	0.059	0.226	0.027	0.116	0.217	0.091

Table 22. Average Decompression Time (s) for JSON Data Transform Variation

	JSON - Average Decompression Times (s)					
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Tags_Lowercase	0.067	0.234	0.025	0.106	0.182	0.087
Tags_LowerNumbers	0.065	0.241	0.032	0.110	0.185	0.091
Tags_LowerUpper	0.063	0.234	0.030	0.105	0.182	0.088
Tags_Uppercase	0.063	0.236	0.032	0.105	0.182	0.087
Tags_UpperLower	0.063	0.234	0.027	0.105	0.182	0.087
Tags_UpperNumbers	0.063	0.241	0.029	0.110	0.185	0.089
Caps_ReusedPrefixes	0.078	0.354	0.036	0.161	0.270	0.118
Caps_UnusedPrefixes	0.076	0.321	0.036	0.144	0.252	0.110
CharNgrams_Existing_Caps	0.058	0.227	0.029	0.123	0.225	0.091
CharNgrams_Existing_Lowercase	0.058	0.226	0.029	0.123	0.225	0.091
CharNgrams_Existing_LowerNumber	0.058	0.225	0.028	0.124	0.224	0.093
CharNgrams_Existing_Numbers	0.057	0.219	0.027	0.120	0.220	0.090
CharNgrams_Existing_Unused	0.056	0.214	0.026	0.117	0.217	0.089
CharNgrams_Existing_UpperNumber	0.058	0.226	0.028	0.124	0.224	0.093
CharNgrams_Unused	0.055	0.203	0.027	0.097	0.209	0.085
WordNgrams_Existing_Caps	0.061	0.232	0.027	0.109	0.202	0.090
WordNgrams_Existing_Lowercase	0.061	0.227	0.028	0.109	0.202	0.091
WordNgrams_Existing_LowerNumber	0.055	0.203	0.024	0.112	0.212	0.086
WordNgrams_Existing_Numbers	0.061	0.227	0.027	0.109	0.203	0.090
WordNgrams_Existing_Unused	0.061	0.224	0.028	0.108	0.199	0.089

Table 22. Average Decompression Time (s) for JSON Data Transform Variation

	JSON - Average Decompression Times (s)					
Data Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_Existing_UpperNumber	0.061	0.226	0.026	0.110	0.202	0.092
WordNgrams_Unused	0.060	0.220	0.026	0.093	0.196	0.089
BWT	0.071	0.262	0.024	0.133	0.300	0.094
LZ77	0.070	0.257	0.027	0.132	0.297	0.093
PAQ	0.070	0.258	0.030	0.127	0.300	0.093
PPM	0.070	0.262	0.027	0.128	0.300	0.093
Untransformed	0.074	0.324	0.016	0.136	0.340	0.099
Average Total	0.059	0.226	0.027	0.116	0.217	0.091

Table 23. Average Decompression Time (s) for JSON CharNgram Level

JSON - Average Decompression Times (s)						
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	0.061	0.257	0.031	0.140	0.246	0.098
CharNgrams_3	0.058	0.228	0.033	0.124	0.229	0.092
CharNgrams_4	0.057	0.222	0.027	0.120	0.223	0.091
CharNgrams_5	0.057	0.216	0.026	0.117	0.219	0.089
CharNgrams_6	0.057	0.212	0.029	0.114	0.216	0.089
CharNgrams_7	0.056	0.211	0.027	0.113	0.215	0.088
CharNgrams_8	0.056	0.210	0.027	0.112	0.212	0.088
CharNgrams_9	0.057	0.212	0.026	0.114	0.213	0.088
CharNgrams_10	0.056	0.211	0.023	0.114	0.213	0.088
Average Total	0.057	0.220	0.028	0.119	0.221	0.090

Table 24. Average Decompression Time (s) for JSON WordNgram Level

JSON - Average Decompression Times (s)						
WordNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
WordNgrams_1	0.059	0.211	0.023	0.100	0.202	0.087
WordNgrams_2	0.060	0.223	0.026	0.109	0.200	0.090
WordNgrams_3	0.060	0.217	0.031	0.102	0.198	0.088
WordNgrams_4	0.061	0.232	0.027	0.114	0.207	0.092
WordNgrams_5	0.061	0.230	0.025	0.111	0.204	0.091
Average Total	0.060	0.223	0.027	0.107	0.202	0.090

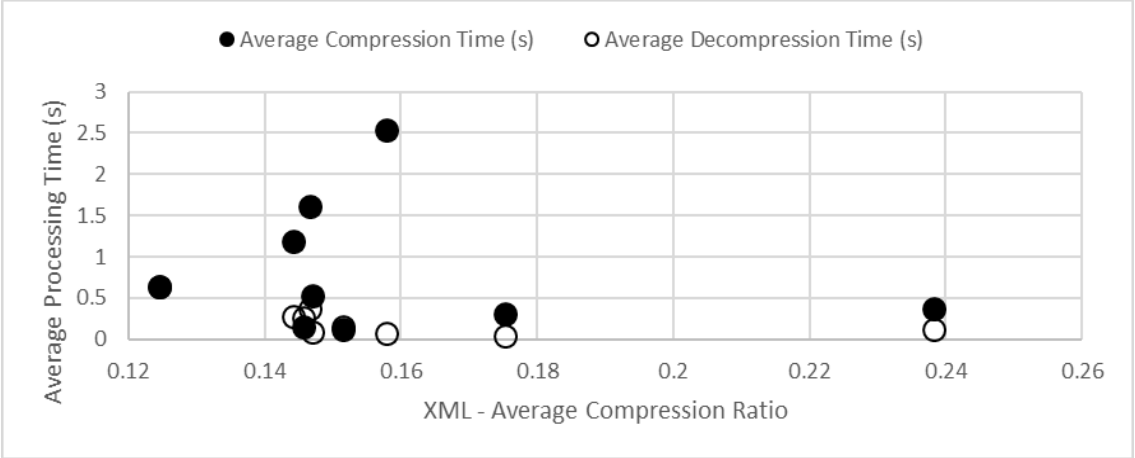

Figure 16. Overall Average Compression and Decompression Times (s) vs Compression Ratios for XML Compression Algorithms

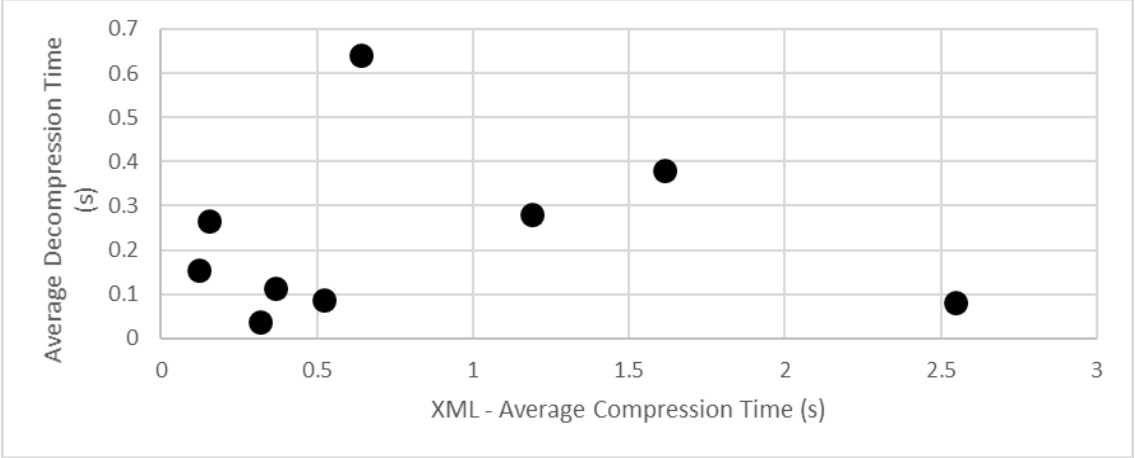

Figure 17. Overall Average Decompression Times (s) vs Compression Times (s) for XML Compression Algorithms

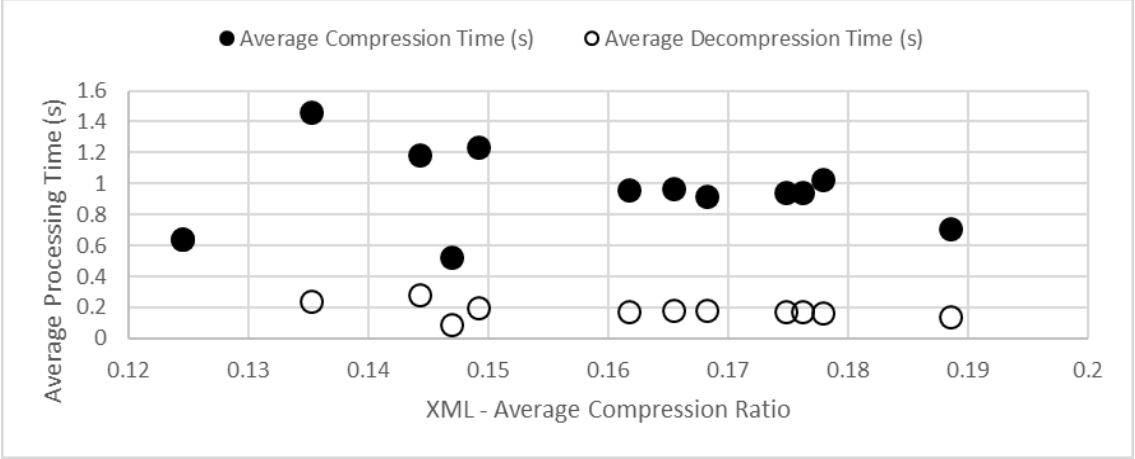

Figure 18. Overall Average Compression and Decompression Times (s) vs Compression Ratios for XML Data Transforms

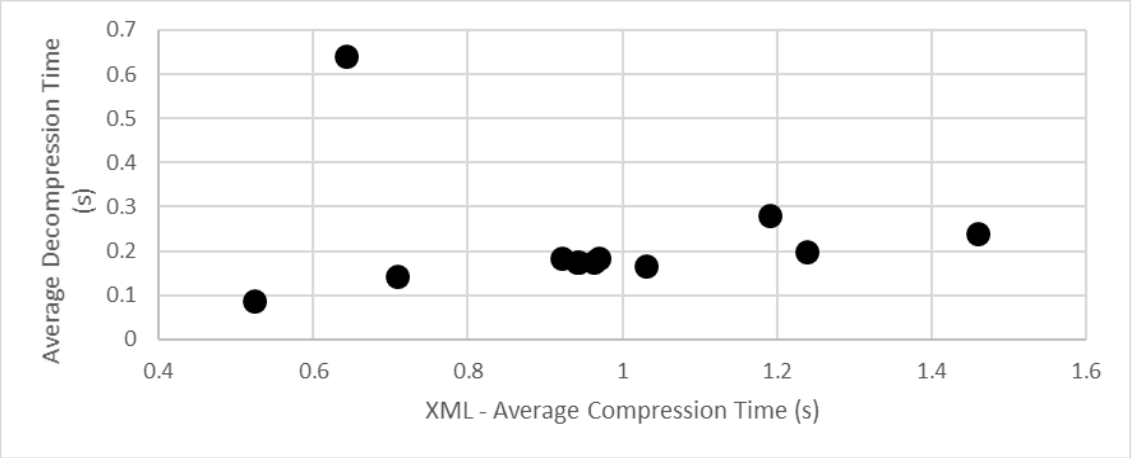

Figure 19. Overall Average Decompression Times (s) vs Compression Times (s) for XML Data Transforms

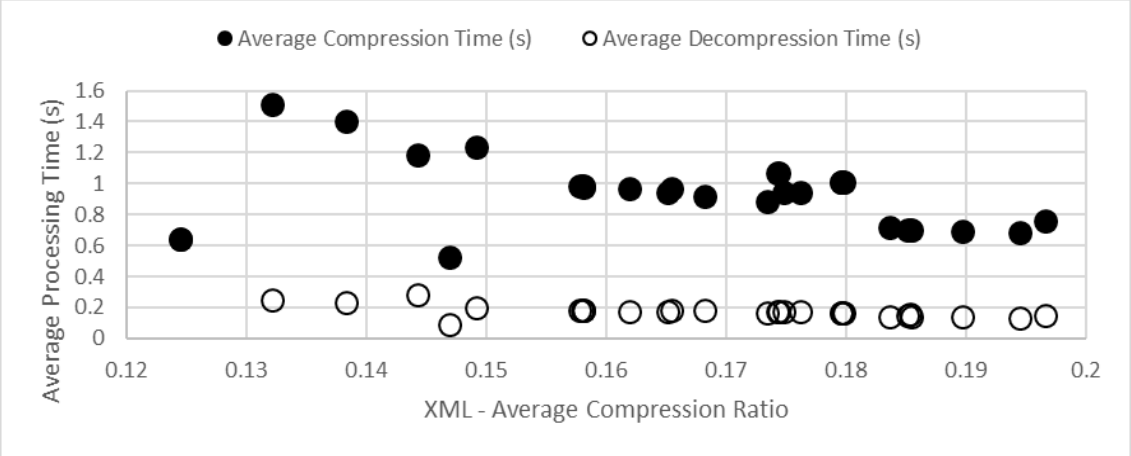

Figure 20. Overall Average Compression and Decompression Times (s) vs Compression Ratios for XML Data Transform Variations

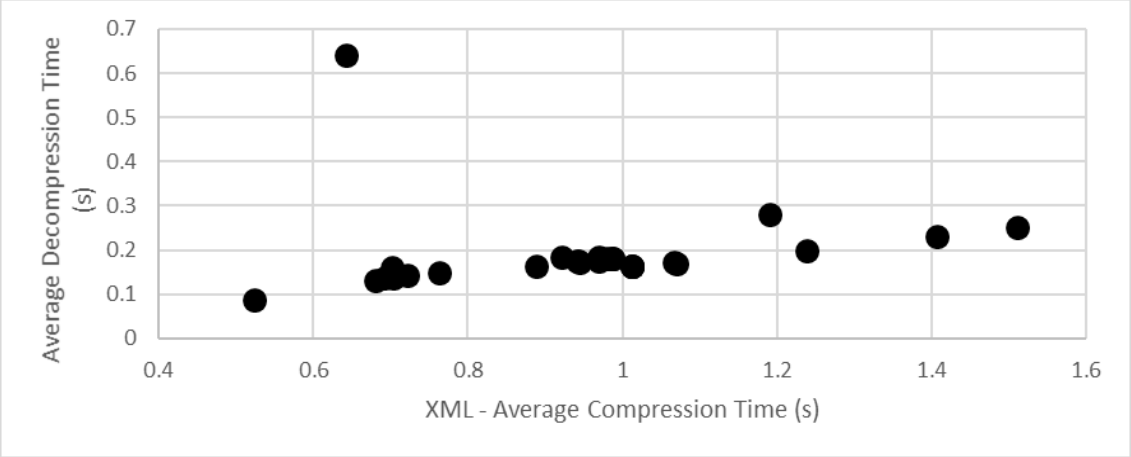

Figure 21. Overall Average Decompression Times (s) vs Compression Times (s) for XML Data Transform Variations

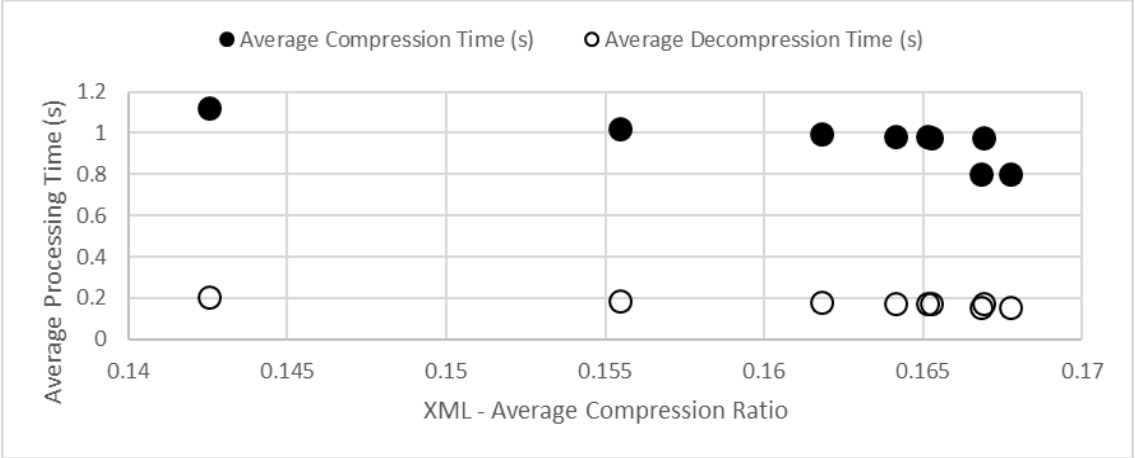

Figure 22. Overall Average Compression and Decompression Times (s) vs Compression Ratios for XML CharNgram Levels

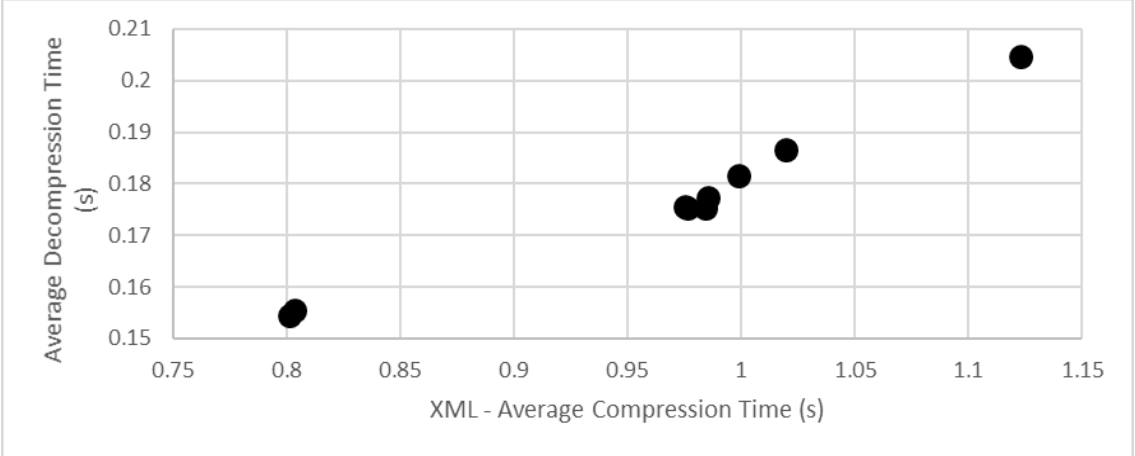

Figure 23. Overall Average Decompression Times (s) vs Compression Times (s) for XML CharNgram Levels

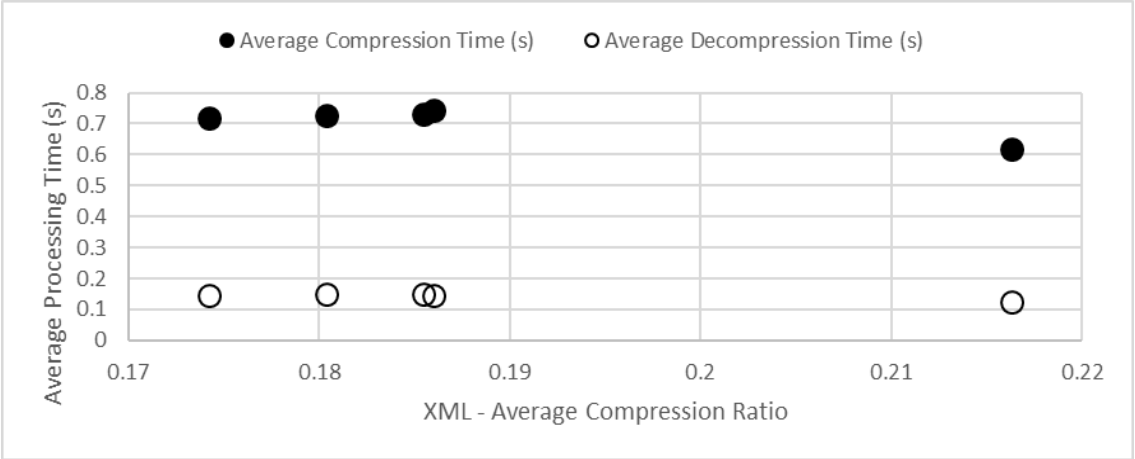

Figure 24. Overall Average Compression and Decompression Times (s) vs Compression Ratios for XML WordNgram Levels

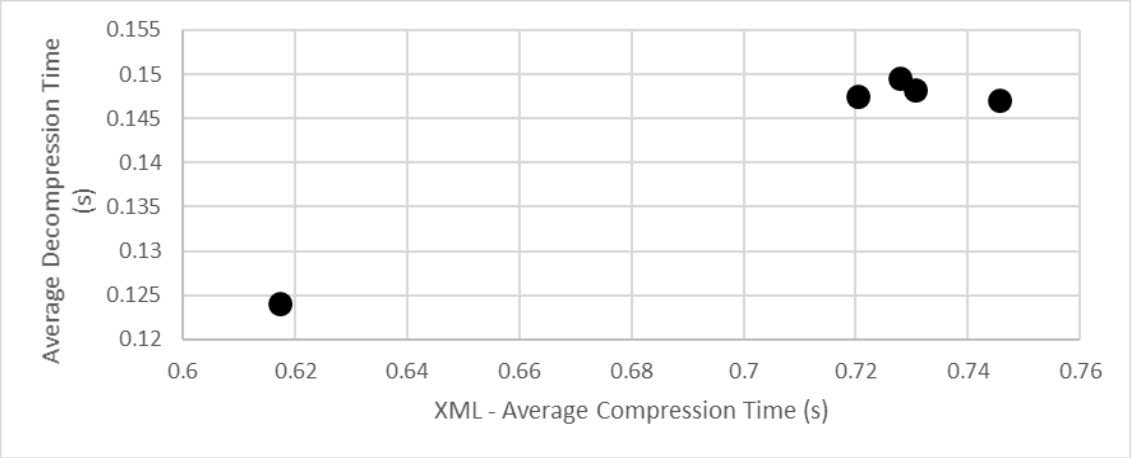

Figure 25. Overall Average Decompression Times (s) vs Compression Times (s) for XML WordNgram Levels

Table 25. Average Compression Results for XML Compression Algorithm

XML Compression Algorithm	Compression Ratio	Compression Time (s)	Decompression Time (s)
7Zip	0.158	2.545	0.082
BZip2	0.147	1.615	0.380
GZip	0.175	0.318	0.037
PPMd	0.152	0.120	0.154
PPMVC	0.146	0.154	0.266
ZPAQ	0.238	0.366	0.114
XMill-BZip2	0.144	1.190	0.282
XMill-GZip	0.147	0.524	0.088
XMill-PPMdi	0.124	0.642	0.642

Table 26. Average Compression Results for XML Data Transform

XML Data Transform	Compression Ratio	Compression Time (s)	Decompression Time (s)
Tags	0.178	1.031	0.166
Caps	0.135	1.459	0.241
CharNgrams	0.162	0.963	0.176
WordNgrams	0.188	0.708	0.143
WRT-BWT	0.165	0.969	0.184
WRT-LZ77	0.168	0.921	0.183
WRT-PAQ	0.176	0.942	0.176
WRT-PPM	0.175	0.942	0.176
Untransformed	0.149	1.238	0.199
XMill-BZip2	0.144	1.190	0.282
XMill-GZip	0.147	0.524	0.088
XMill-PPMdi	0.124	0.642	0.642

Table 27. Average Compression Results for XML Data Transform Variation

XML			
Data Transform Variation	Compression Ratio	Compression Time (s)	Decompression Time (s)
Tags_Lowercase	0.180	1.011	0.164
Tags_LowerNumbers	0.174	1.067	0.172
Tags_LowerUpper	0.180	1.013	0.163
Tags_Uppercase	0.180	1.011	0.164
Tags_UpperLower	0.180	1.013	0.164
Tags_UpperNumbers	0.174	1.069	0.170
Caps_ReusedPrefixes	0.132	1.511	0.251
Caps_UnusedPrefixes	0.138	1.407	0.231
CharNgrams_Existing_Caps	0.158	0.985	0.180
CharNgrams_Existing_Lowercase	0.158	0.987	0.181
CharNgrams_Existing_LowerNumber	0.158	0.978	0.181
CharNgrams_Existing_Numbers	0.162	0.969	0.175
CharNgrams_Existing_Unused	0.165	0.945	0.172
CharNgrams_Existing_UpperNumber	0.158	0.988	0.182
CharNgrams_Unused	0.173	0.889	0.163
WordNgrams_Existing_Caps	0.185	0.702	0.161
WordNgrams_Existing_Lowercase	0.185	0.703	0.137
WordNgrams_Existing_LowerNumber	0.184	0.721	0.142
WordNgrams_Existing_Numbers	0.197	0.762	0.150
WordNgrams_Existing_Unused	0.190	0.691	0.136
WordNgrams_Existing_UpperNumber	0.185	0.704	0.144

Table 27. Average Compression Results for XML Data Transform Variation

XML			
Data Transform Variation	Compression Ratio	Compression Time (s)	Decompression Time (s)
WordNgrams_Unused	0.195	0.681	0.132
WRT-BWT	0.165	0.969	0.184
WRT-LZ77	0.168	0.921	0.183
WRT-PAQ	0.176	0.942	0.176
WRT-PPM	0.175	0.942	0.176
Untransformed	0.149	1.238	0.199
XMill-BZip2	0.144	1.190	0.282
XMill-GZip	0.147	0.524	0.088
XMill-PPMdi	0.124	0.642	0.642

Table 28. Average Compression Results for XML CharNgram Level

XML CharNgram Level	Compression Ratio	Compression Time (s)	Decompression Time (s)
CharNgrams_2	0.143	1.123	0.205
CharNgrams_3	0.155	1.020	0.187
CharNgrams_4	0.162	0.999	0.182
CharNgrams_5	0.164	0.985	0.177
CharNgrams_6	0.165	0.975	0.176
CharNgrams_7	0.165	0.984	0.175
CharNgrams_8	0.167	0.977	0.175
CharNgrams_9	0.168	0.803	0.155
CharNgrams_10	0.167	0.801	0.155

Table 29. Average Compression Results for XML WordNgram Level

XML WordNgram Level	Compression Ratio	Compression Time (s)	Decompression Time (s)
WordNgrams_1	0.174	0.720	0.148
WordNgrams_2	0.216	0.617	0.124
WordNgrams_3	0.186	0.746	0.147
WordNgrams_4	0.185	0.730	0.148
WordNgrams_5	0.180	0.728	0.149

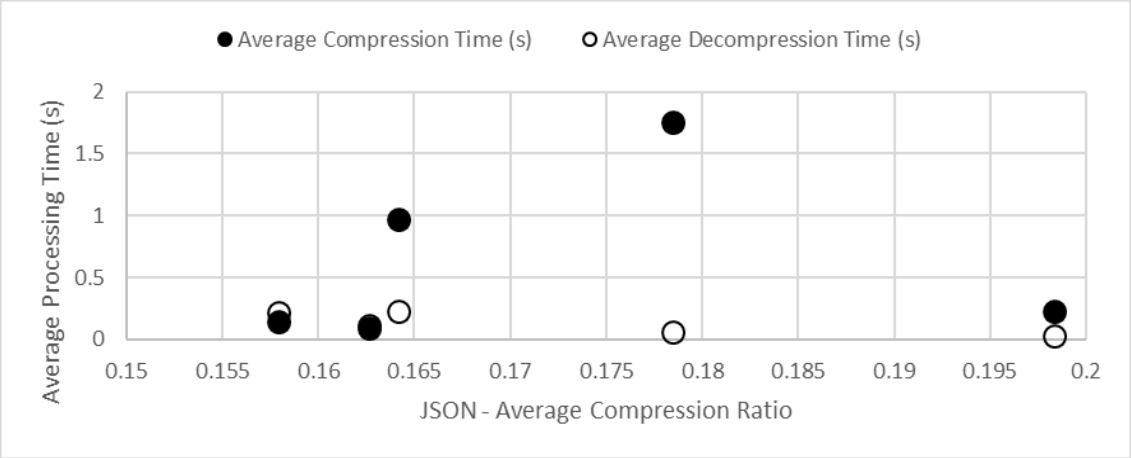

Figure 26. Overall Average Compression and Decompression Times (s) vs Compression Ratios for JSON Compression Algorithms

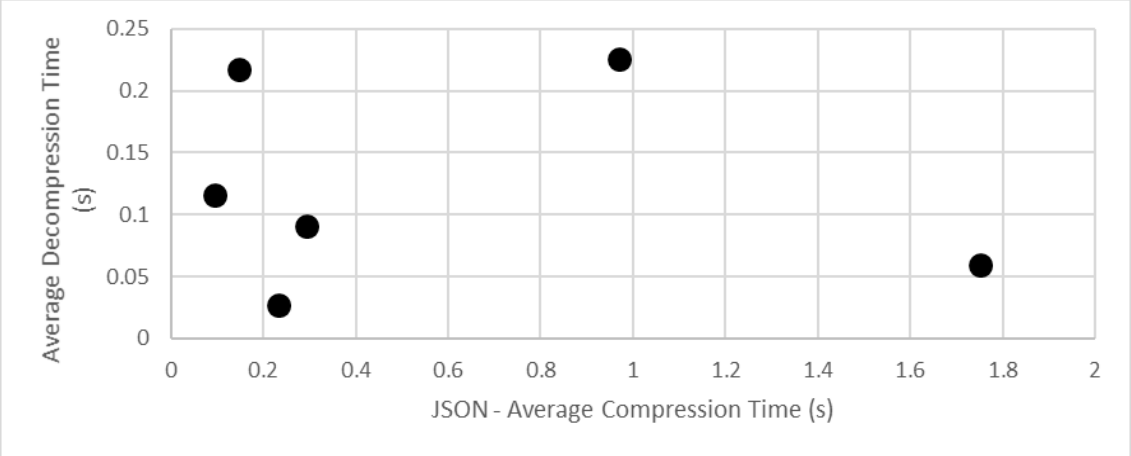

Figure 27. Overall Average Decompression Times (s) vs Compression Times (s) for JSON Compression Algorithms

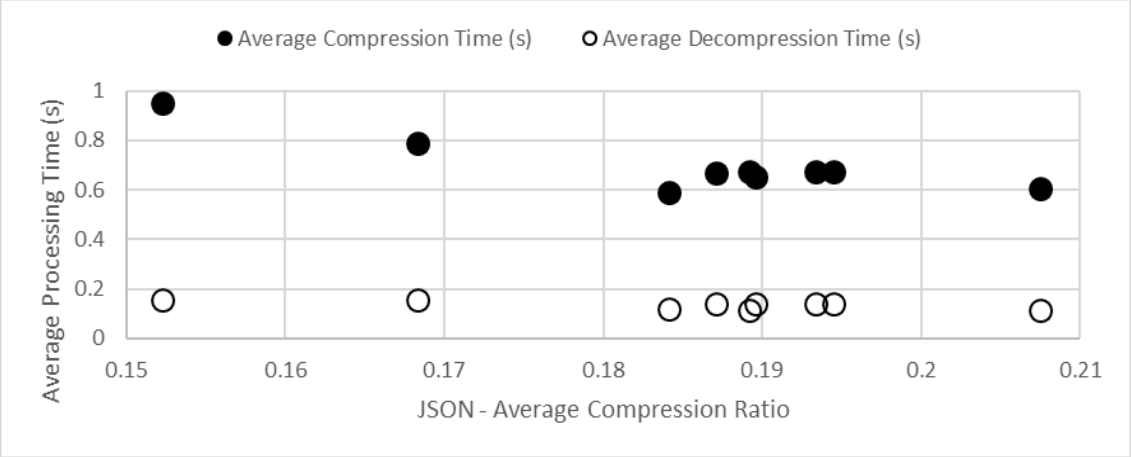

Figure 28. Overall Average Compression and Decompression Times (s) vs Compression Ratios for JSON Data Transforms

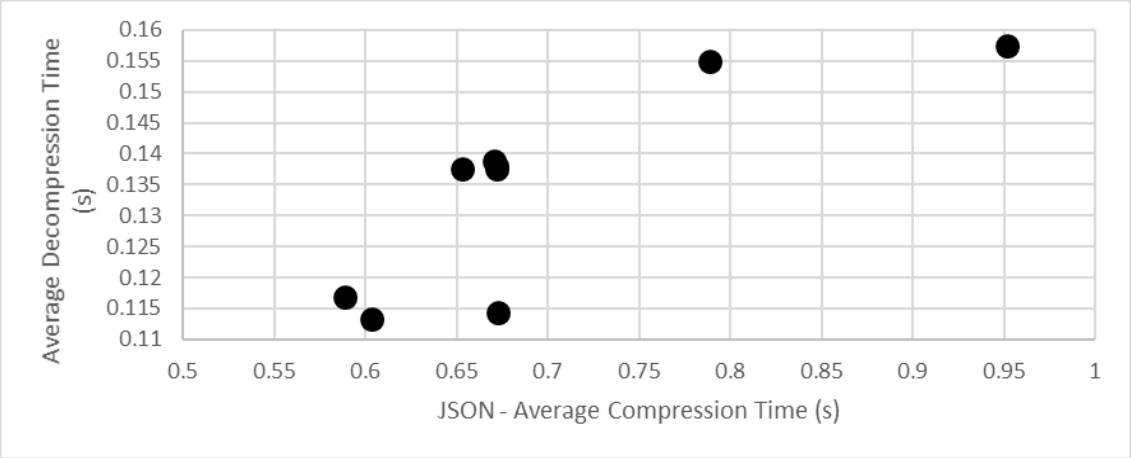

Figure 29. Overall Average Decompression Times (s) vs Compression Times (s) for JSON Data Transforms

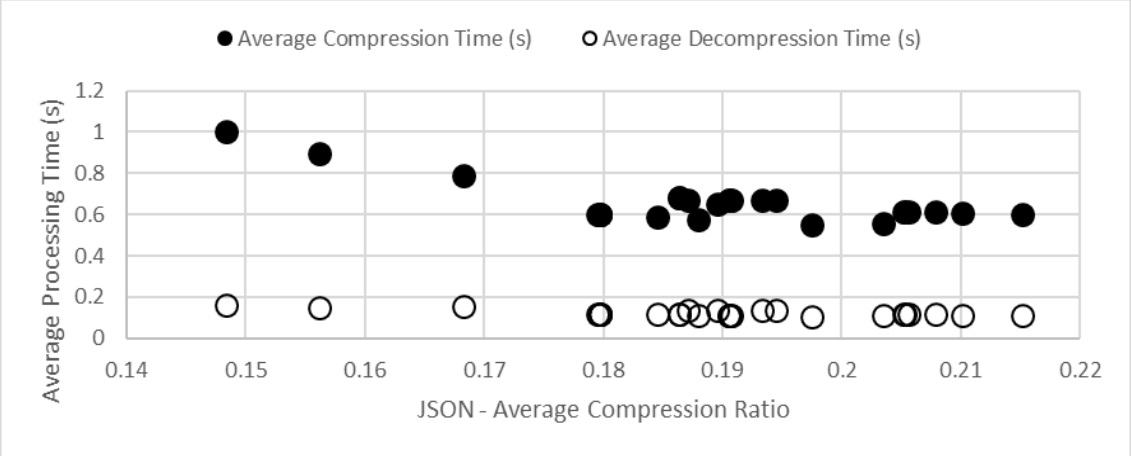

Figure 30. Overall Average Compression and Decompression Times (s) vs Compression Ratios for JSON Data Transform Variations

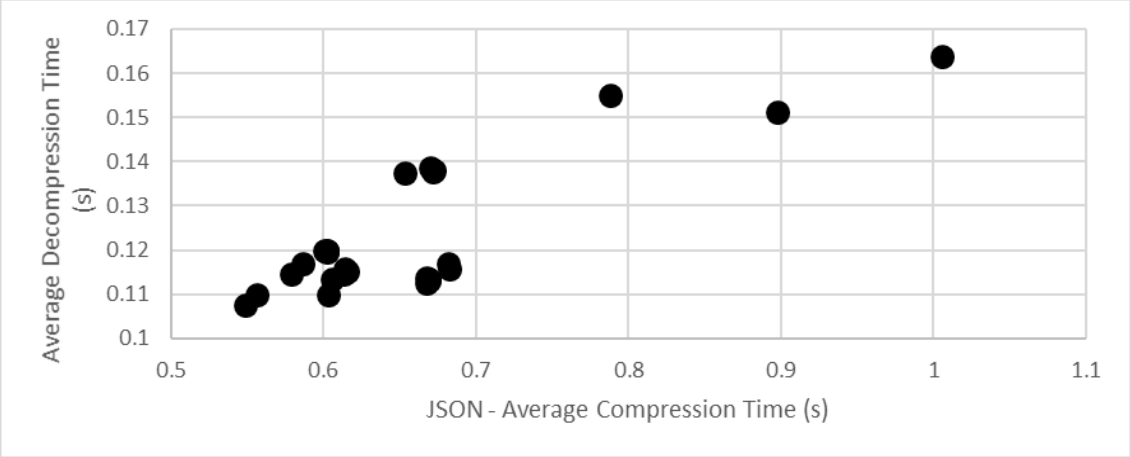

Figure 31. Overall Average Decompression Times (s) vs Compression Times (s) for JSON Data Transform Variations

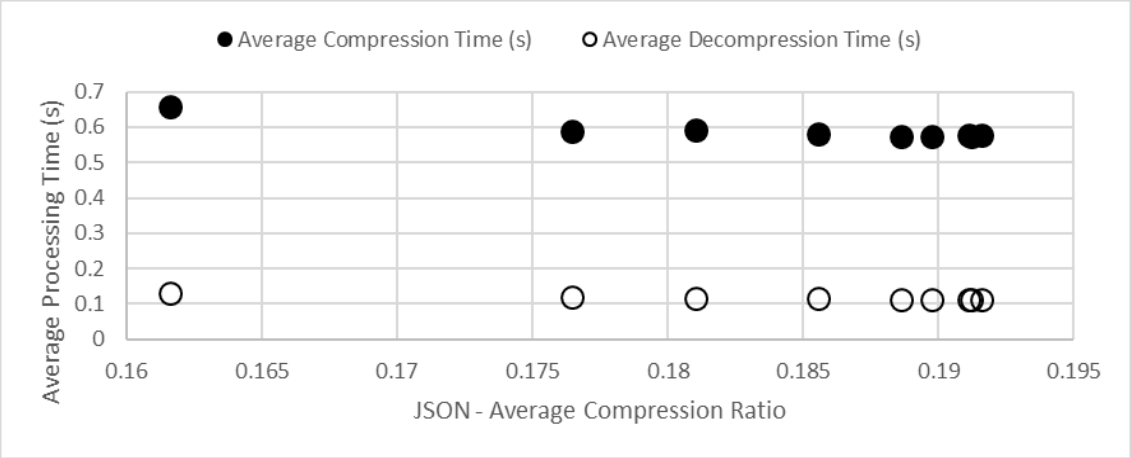

Figure 32. Overall Average Compression and Decompression Times (s) vs Compression Ratios for JSON CharNgram Levels

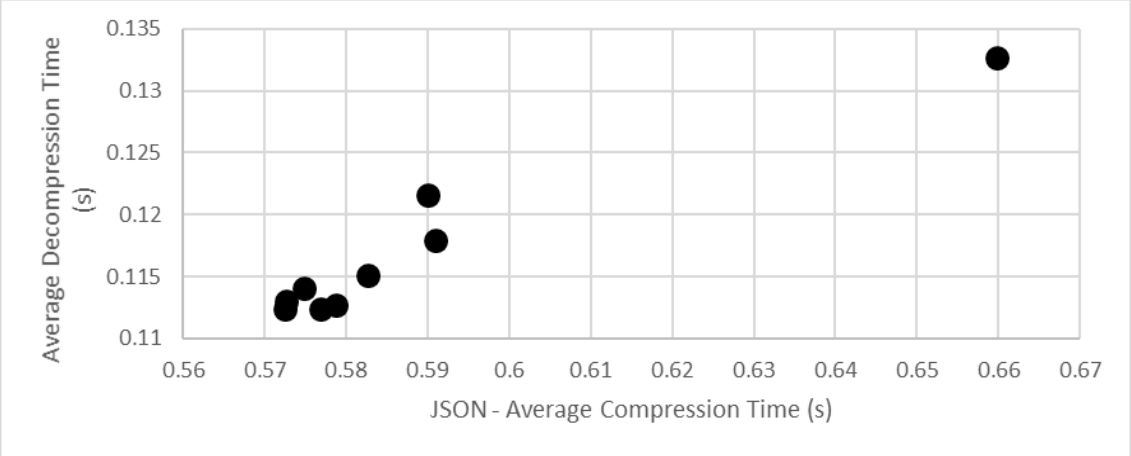

Figure 33. Overall Average Decompression Times (s) vs Compression Times (s) for JSON CharNgram Levels

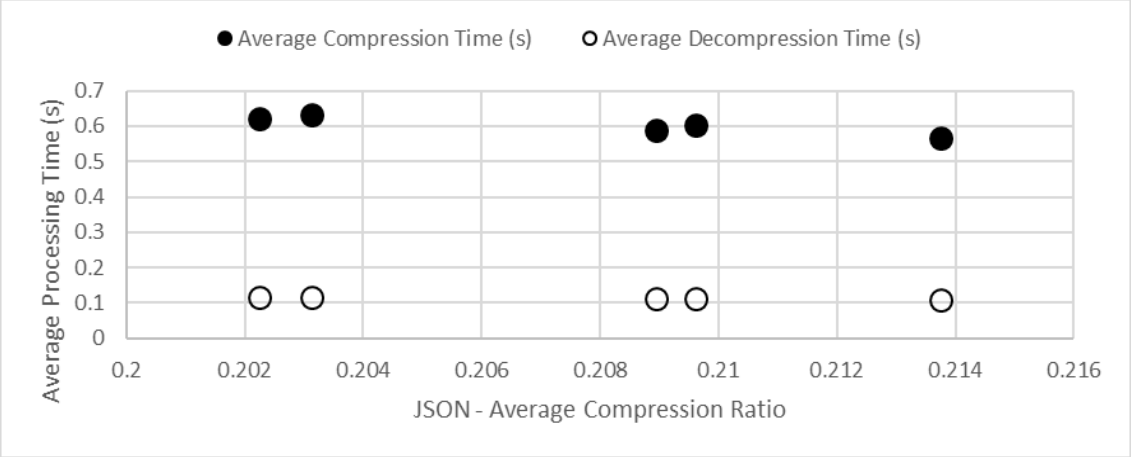

Figure 34. Overall Average Compression and Decompression Times (s) vs Compression Ratios for JSON WordNgram Levels

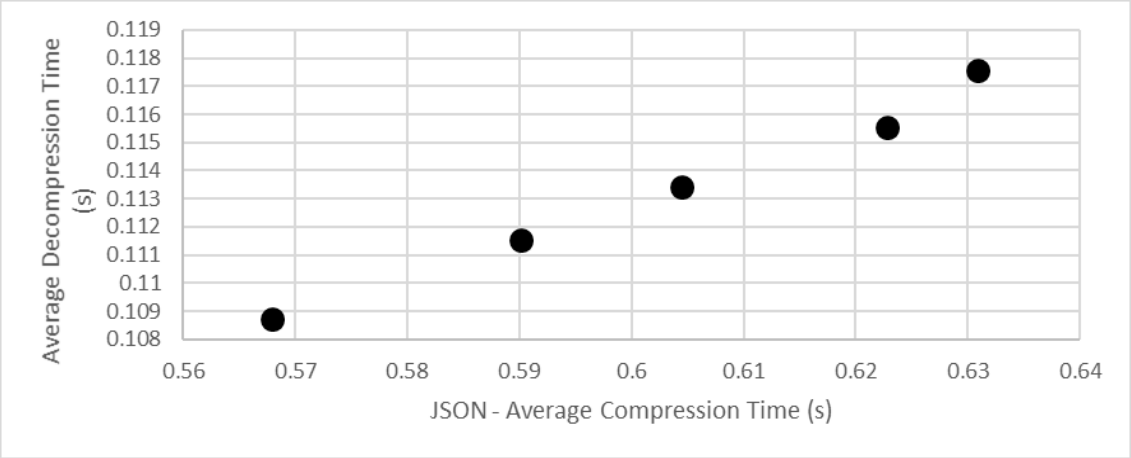

Figure 35. Overall Average Decompression Times (s) vs Compression Times (s) for JSON WordNgram Levels

Table 30. Average Compression Results for JSON Compression Algorithm

JSON			
Compression Algorithm	Compression Ratio	Compression Time (s)	Decompression Time (s)
7Zip	0.178	1.751	0.059
BZip2	0.164	0.970	0.226
GZip	0.198	0.231	0.027
PPMd	0.163	0.094	0.116
PPMVC	0.158	0.146	0.217
ZPAQ	0.274	0.294	0.091

Table 31. Average Compression Results for JSON Data Transform

JSON Data Transform	Compression Ratio	Compression Time (s)	Decompression Time (s)
Tags	0.189	0.673	0.114
Caps	0.152	0.951	0.157
CharNgrams	0.184	0.589	0.117
WordNgrams	0.208	0.603	0.113
WRT-BWT	0.187	0.670	0.139
WRT-LZ77	0.190	0.653	0.138
WRT-PAQ	0.194	0.672	0.138
WRT-PPM	0.193	0.672	0.138
Untransformed	0.168	0.788	0.155

Table 32. Average Compression Results for JSON Data Transform Variation

JSON			
Data Transform Variation	Compression Ratio	Compression Time (s)	Decompression Time (s)
Tags_Lowercase	0.191	0.669	0.113
Tags_LowerNumbers	0.186	0.682	0.117
Tags_LowerUpper	0.191	0.669	0.113
Tags_Uppercase	0.191	0.667	0.114
Tags_UpperLower	0.191	0.668	0.113
Tags_UpperNumbers	0.186	0.683	0.116
Caps_ReusedPrefixes	0.148	1.006	0.164
Caps_UnusedPrefixes	0.156	0.897	0.151
CharNgrams_Existing_Caps	0.180	0.602	0.120
CharNgrams_Existing_Lowercase	0.180	0.602	0.120
CharNgrams_Existing_LowerNumber	0.180	0.600	0.120
CharNgrams_Existing_Numbers	0.185	0.586	0.117
CharNgrams_Existing_Unused	0.188	0.578	0.114
CharNgrams_Existing_UpperNumber	0.180	0.602	0.120
CharNgrams_Unused	0.197	0.549	0.107
WordNgrams_Existing_Caps	0.205	0.614	0.116
WordNgrams_Existing_Lowercase	0.205	0.616	0.115
WordNgrams_Existing_LowerNumber	0.204	0.557	0.110
WordNgrams_Existing_Numbers	0.208	0.613	0.115
WordNgrams_Existing_Unused	0.210	0.606	0.113
WordNgrams_Existing_UpperNumber	0.206	0.614	0.115

Table 32. Average Compression Results for JSON Data Transform Variation

JSON			
Data Transform Variation	Compression Ratio	Compression Time (s)	Decompression Time (s)
WordNgrams_Unused	0.215	0.603	0.110
WRT-BWT	0.187	0.670	0.139
WRT-LZ77	0.190	0.653	0.138
WRT-PAQ	0.194	0.672	0.138
WRT-PPM	0.193	0.672	0.138
Untransformed	0.168	0.788	0.155

Table 33. Average Compression Results for JSON CharNgram Level

JSON CharNgram Level	Compression Ratio	Compression Time (s)	Decompression Time (s)
CharNgrams_2	0.162	0.660	0.133
CharNgrams_3	0.176	0.590	0.122
CharNgrams_4	0.181	0.591	0.118
CharNgrams_5	0.186	0.583	0.115
CharNgrams_6	0.189	0.575	0.114
CharNgrams_7	0.191	0.573	0.113
CharNgrams_8	0.190	0.572	0.112
CharNgrams_9	0.191	0.579	0.113
CharNgrams_10	0.192	0.577	0.112

Table 34. Average Compression Results for JSON WordNgram Level

JSON WordNgram Level	Compression Ratio	Compression Time (s)	Decompression Time (s)
WordNgrams_1	0.214	0.568	0.109
WordNgrams_2	0.210	0.604	0.113
WordNgrams_3	0.209	0.590	0.112
WordNgrams_4	0.203	0.631	0.118
WordNgrams_5	0.202	0.623	0.116

Appendix E: SMILES Overall Result Graphs and Tables

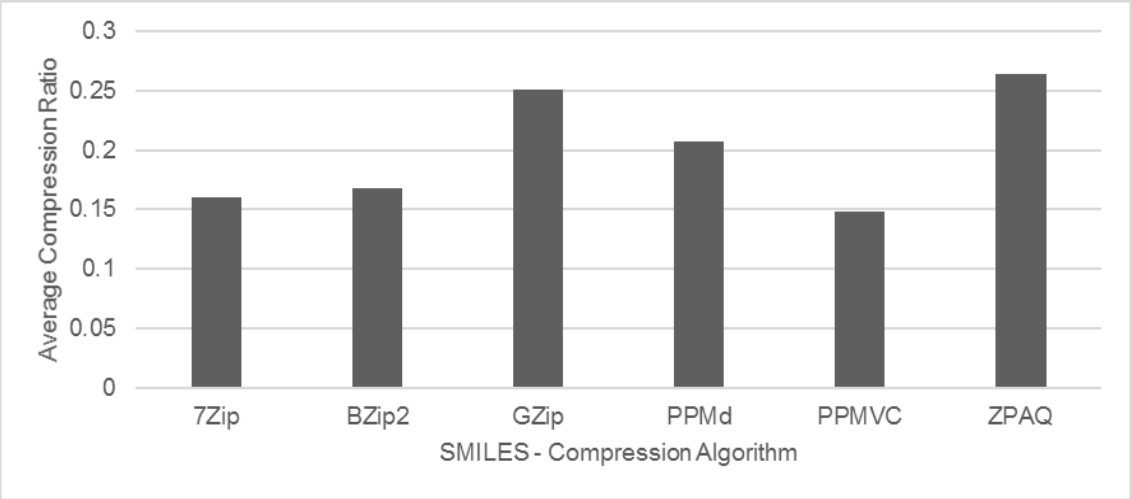

Figure 1. Overall Average Compression Ratios for SMILES Compression Algorithms

Figure 2. Overall Average Compression Ratios for SMILES Data Transforms

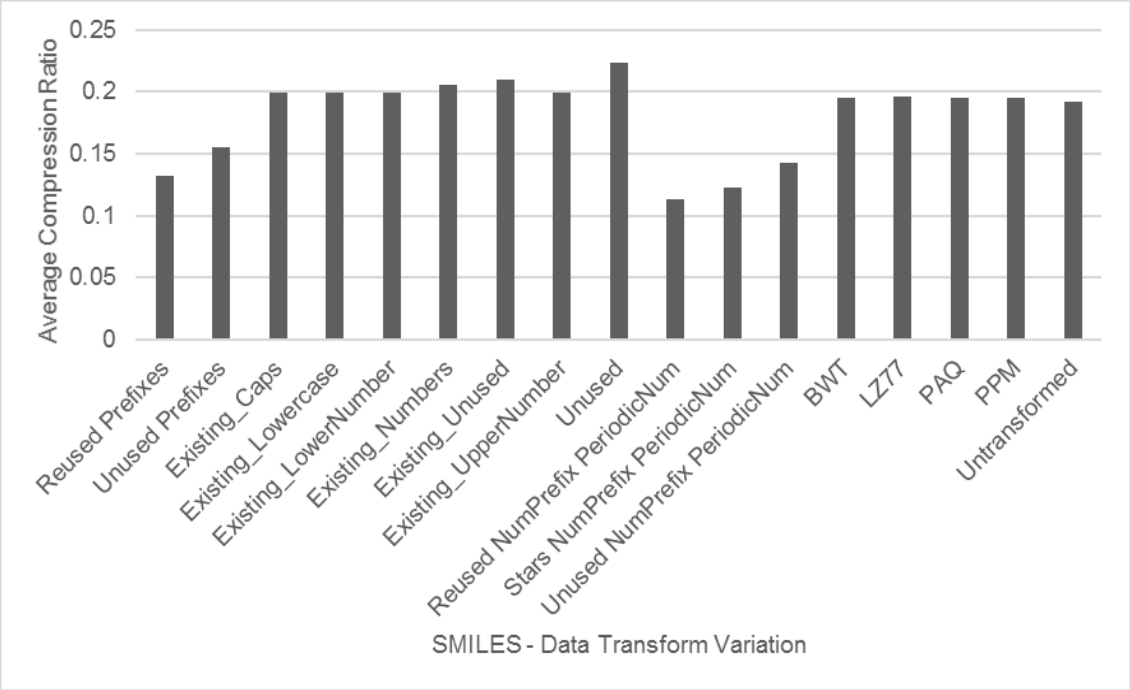

Figure 3. Overall Average Compression Ratios for SMILES Data Transform Variations

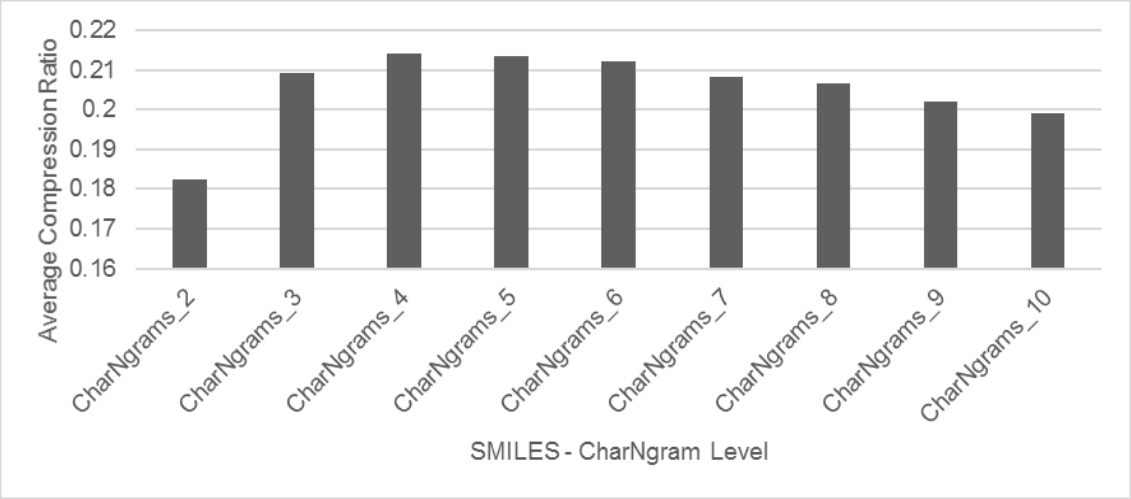

Figure 4. Overall Average Compression Ratios for SMILES CharNgram Levels

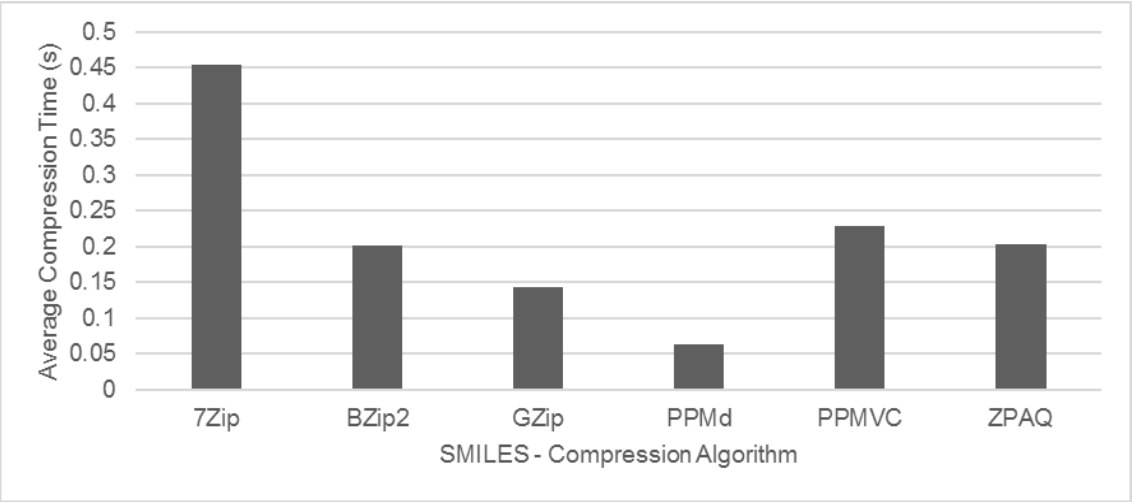

Figure 5. Overall Average Compression Times (s) for SMILES Compression Algorithms

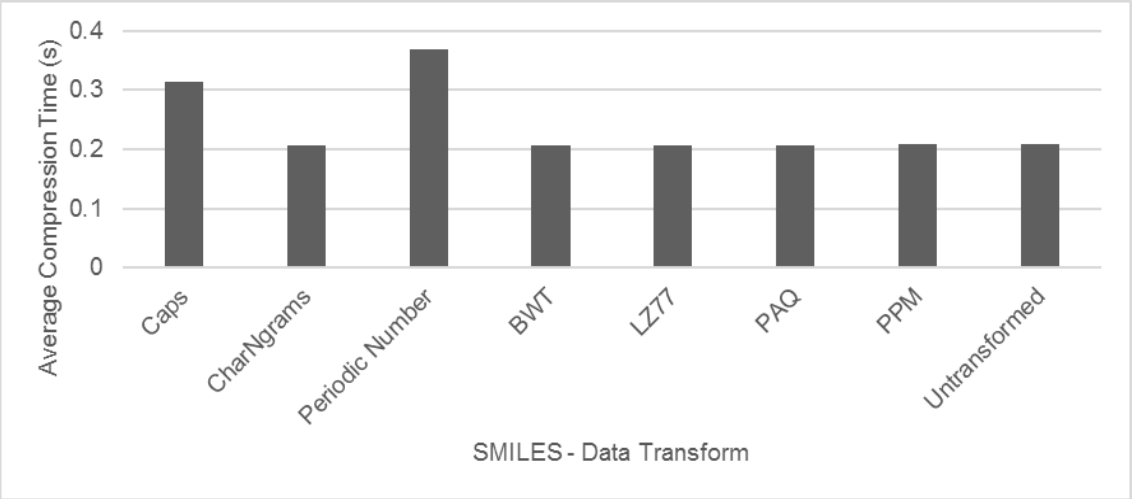

Figure 6. Overall Average Compression Times (s) for SMILES Data Transforms

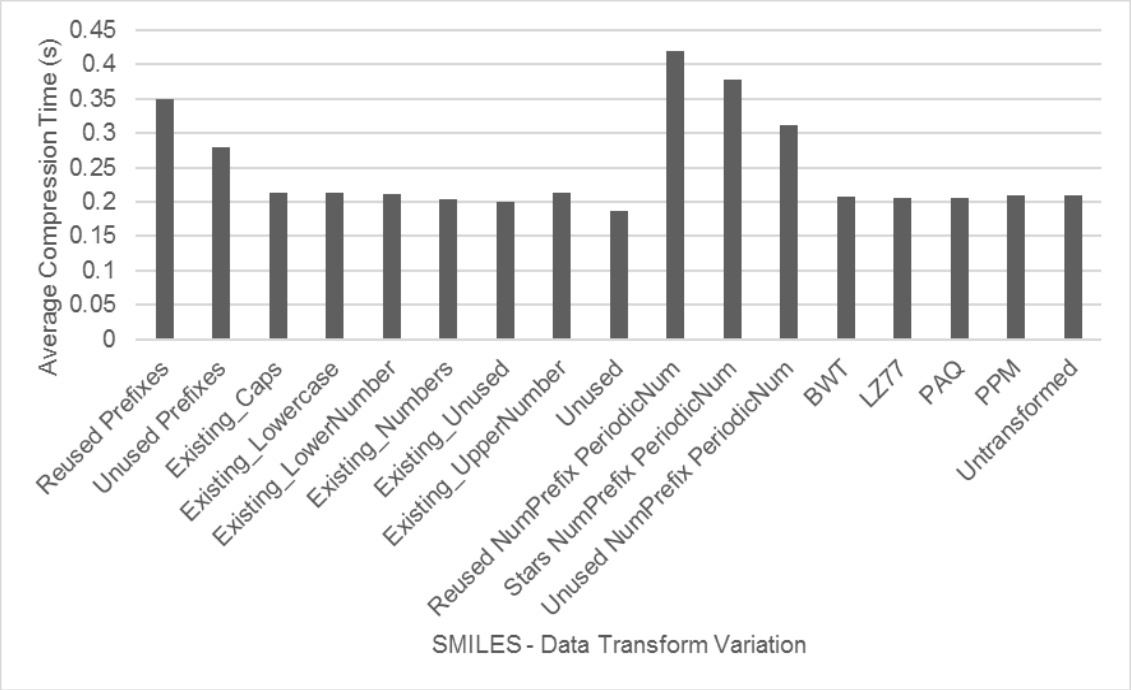

Figure 7. Overall Average Compression Times (s) for SMILES Data Transform Variations

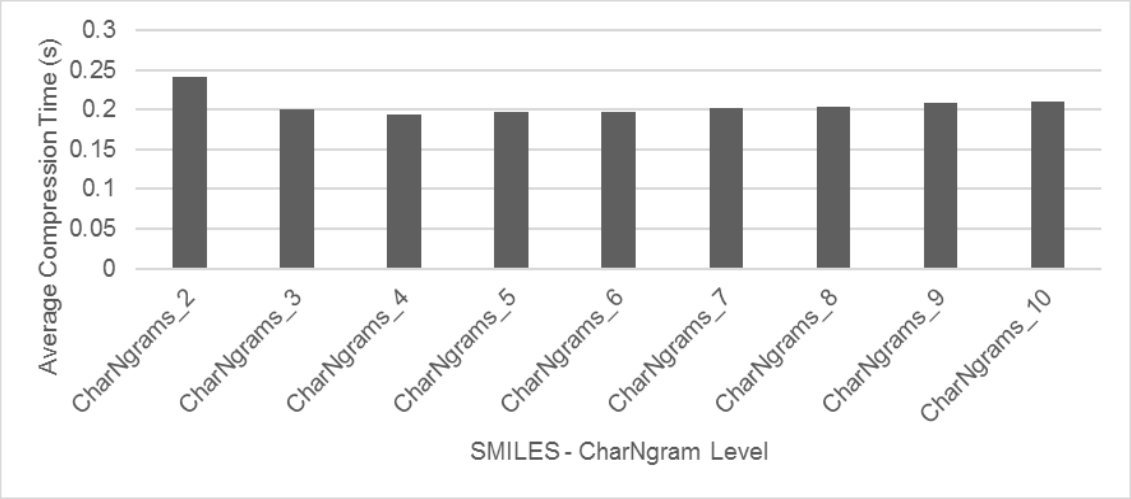

Figure 8. Overall Average Compression Times (s) for SMILES CharNgram Levels

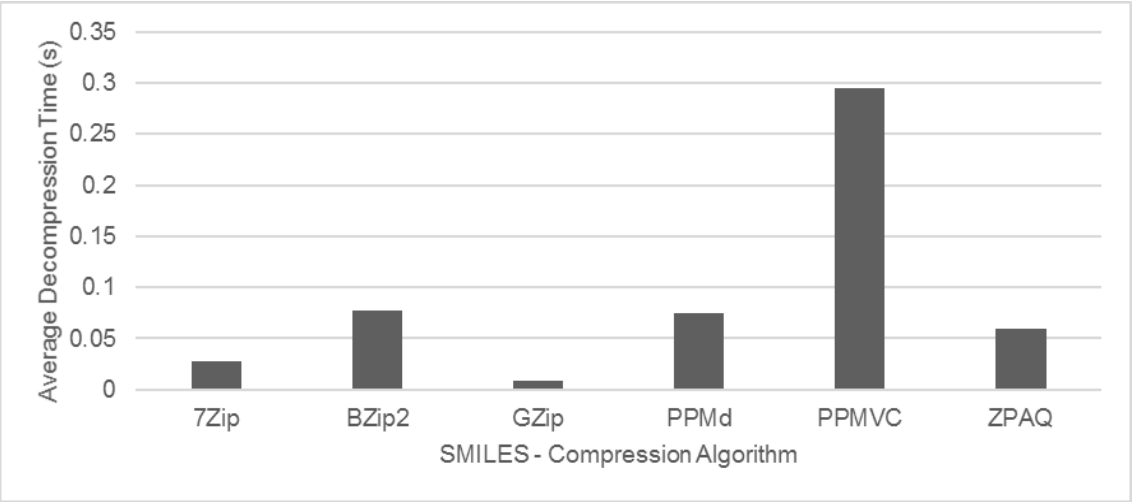

Figure 9. Overall Average Decompression Times (s) for SMILES Compression Algorithms

Figure 10. Overall Average Decompression Times (s) for SMILES Data Transforms

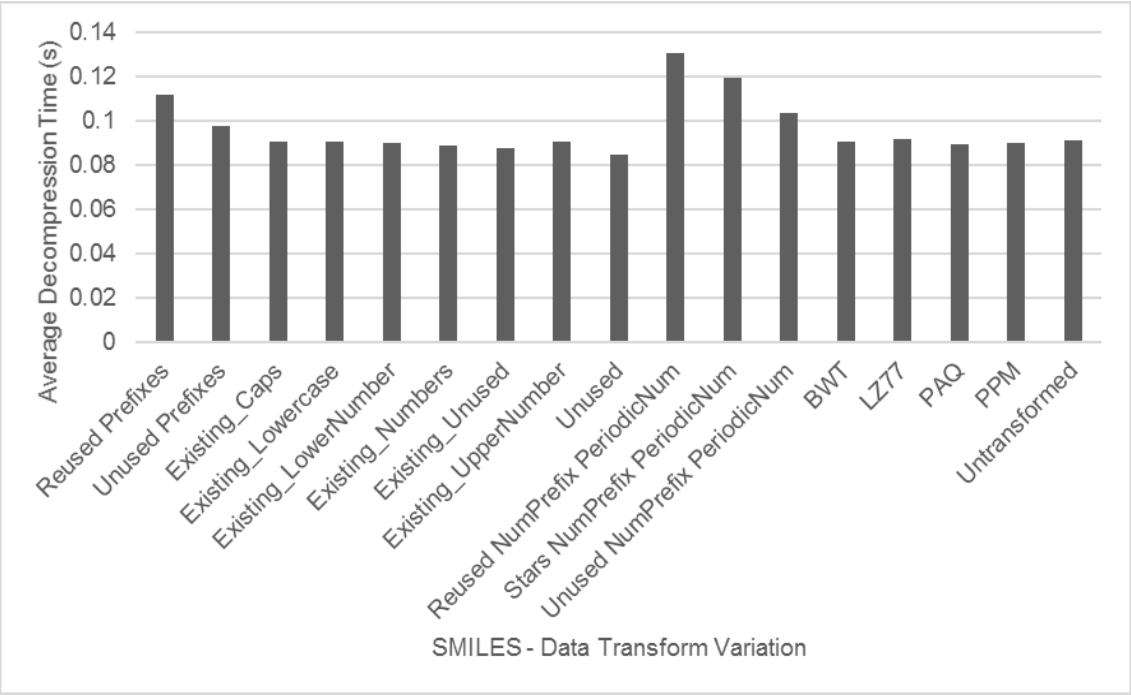

Figure 11. Overall Average Decompression Times (s) for SMILES Data Transform Variations

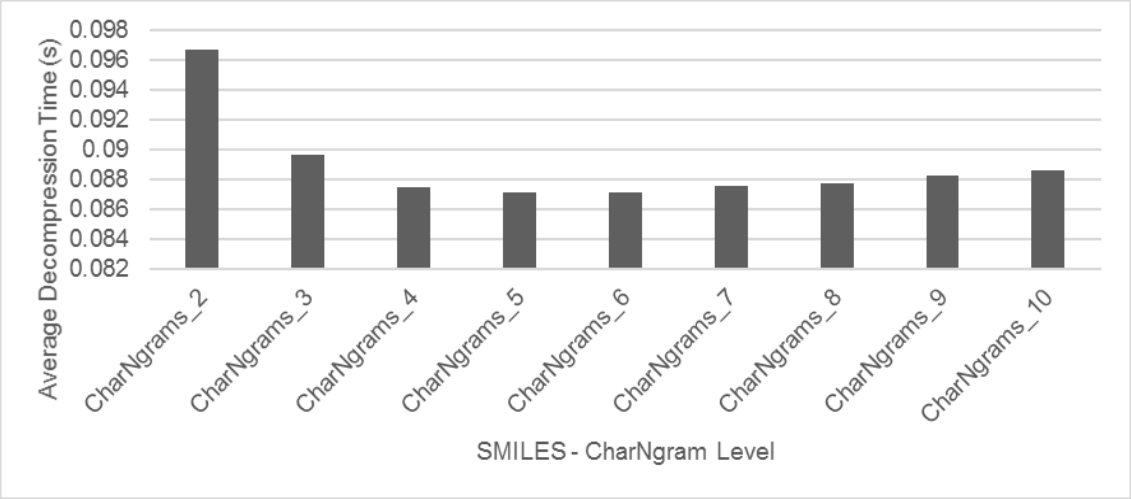

Figure 12. Overall Average Decompression Times (s) for SMILES CharNgram Levels

Table 1. Average Compression Ratios for SMILES Data Transforms

	SMILES - Average Compression Ratio					
Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Caps	0.107	0.110	0.180	0.149	0.104	0.212
CharNgrams	0.166	0.174	0.258	0.212	0.152	0.270
Periodic Number	0.094	0.093	0.158	0.129	0.090	0.194
BWT	0.155	0.164	0.242	0.207	0.148	0.258
LZ77	0.155	0.165	0.242	0.207	0.148	0.258
PAQ	0.155	0.164	0.242	0.207	0.148	0.258
PPM	0.155	0.164	0.242	0.207	0.148	0.258
Untransformed	0.151	0.161	0.238	0.201	0.143	0.255
Total Average	0.161	0.168	0.251	0.207	0.148	0.264

Table 2. Average Compression Ratios for SMILES Data Transform Variations

	SMILES - Average Compression Ratio					
Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Reused Prefixes	0.097	0.098	0.166	0.137	0.094	0.202
Unused Prefixes	0.117	0.122	0.194	0.161	0.113	0.222
Existing_Caps	0.160	0.167	0.251	0.207	0.147	0.266
Existing_Lowercase	0.160	0.167	0.251	0.206	0.147	0.266
Existing_LowerNum	0.160	0.167	0.250	0.207	0.147	0.264
Existing_Numbers	0.166	0.175	0.258	0.214	0.153	0.270
Existing_Unused	0.171	0.178	0.265	0.217	0.156	0.274
Existing_UpperNum	0.160	0.167	0.250	0.207	0.147	0.264
Unused	0.185	0.194	0.281	0.228	0.168	0.284
Reused NumPrefix PeriodicNum	0.083	0.080	0.141	0.114	0.081	0.182
Stars NumPrefix PeriodicNum	0.092	0.089	0.152	0.125	0.087	0.190
Unused NumPrefix PeriodicNum	0.106	0.110	0.179	0.148	0.103	0.212
BWT	0.155	0.164	0.242	0.207	0.148	0.258
LZ77	0.155	0.165	0.242	0.207	0.148	0.258
PAQ	0.155	0.164	0.242	0.207	0.148	0.258
PPM	0.155	0.164	0.242	0.207	0.148	0.258
Untransformed	0.151	0.161	0.238	0.201	0.143	0.255
Total Average	0.161	0.168	0.251	0.207	0.148	0.264

Table 3. Average Compression Ratios for SMILES CharNgram Levels

	SMILES - Average Compression Ratio					
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	0.147	0.150	0.236	0.183	0.131	0.248
CharNgrams_3	0.173	0.176	0.268	0.212	0.154	0.274
CharNgrams_4	0.176	0.182	0.270	0.219	0.158	0.279
CharNgrams_5	0.174	0.183	0.267	0.222	0.159	0.278
CharNgrams_6	0.172	0.181	0.265	0.222	0.158	0.276
CharNgrams_7	0.168	0.177	0.260	0.217	0.155	0.273
CharNgrams_8	0.166	0.175	0.257	0.215	0.154	0.271
CharNgrams_9	0.161	0.171	0.252	0.211	0.151	0.266
CharNgrams_10	0.158	0.168	0.248	0.209	0.149	0.263
Total Average	0.166	0.174	0.258	0.212	0.152	0.270

Table 4. Average Compression Times (s) for SMILES Data Transforms

SMILES - Average Compression Times (s)						
Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Caps	0.812	0.350	0.182	0.070	0.221	0.252
CharNgrams	0.419	0.186	0.139	0.062	0.230	0.199
Periodic Number	0.990	0.437	0.200	0.076	0.237	0.278
BWT	0.431	0.192	0.143	0.060	0.218	0.196
LZ77	0.429	0.192	0.144	0.060	0.216	0.195
PAQ	0.430	0.191	0.143	0.060	0.214	0.197
PPM	0.447	0.192	0.142	0.060	0.215	0.196
Untransformed	0.441	0.192	0.146	0.058	0.216	0.201
Total Average	0.455	0.201	0.143	0.063	0.229	0.204

Table 5. Average Compression Times (s) for SMILES Data Transform Variations

	SMILES - Average Compression Times (s)					
Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Reused Prefixes	0.917	0.416	0.192	0.075	0.227	0.266
Unused Prefixes	0.708	0.285	0.171	0.066	0.214	0.238
Existing_Caps	0.448	0.192	0.147	0.062	0.232	0.203
Existing_Lowercase	0.442	0.192	0.147	0.062	0.232	0.203
Existing_LowerNum	0.443	0.191	0.141	0.062	0.233	0.203
Existing_Numbers	0.413	0.184	0.138	0.061	0.231	0.198
Existing_Unused	0.395	0.182	0.135	0.061	0.228	0.196
Existing_UpperNum	0.445	0.192	0.141	0.062	0.233	0.202
Unused	0.350	0.166	0.124	0.063	0.223	0.191
Reused NumPrefix PeriodicNum	1.159	0.499	0.224	0.082	0.253	0.300
Stars NumPrefix PeriodicNum	0.987	0.479	0.203	0.077	0.240	0.282
Unused NumPrefix PeriodicNum	0.824	0.332	0.173	0.069	0.220	0.251
BWT	0.431	0.192	0.143	0.060	0.218	0.196
LZ77	0.429	0.192	0.144	0.060	0.216	0.195
PAQ	0.430	0.191	0.143	0.060	0.214	0.197
PPM	0.447	0.192	0.142	0.060	0.215	0.196
Untransformed	0.441	0.192	0.146	0.058	0.216	0.201
Total Average	0.455	0.201	0.143	0.063	0.229	0.204

Table 6. Average Compression Times (s) for SMILES CharNgram Levels

SMILES - Average Compression Times (s)						
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	0.551	0.219	0.150	0.069	0.234	0.219
CharNgrams_3	0.392	0.178	0.130	0.065	0.234	0.198
CharNgrams_4	0.370	0.175	0.133	0.062	0.232	0.195
CharNgrams_5	0.385	0.177	0.137	0.060	0.231	0.194
CharNgrams_6	0.386	0.179	0.136	0.061	0.230	0.194
CharNgrams_7	0.405	0.181	0.139	0.060	0.229	0.196
CharNgrams_8	0.412	0.185	0.141	0.060	0.229	0.197
CharNgrams_9	0.430	0.186	0.143	0.060	0.228	0.199
CharNgrams_10	0.444	0.190	0.143	0.060	0.225	0.201
Total Average	0.419	0.186	0.139	0.062	0.230	0.199

Table 7. Average Decompression Times (s) for SMILES Data Transforms

SMILES - Average Decompression Times (s)						
Transform	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Caps	0.031	0.108	0.010	0.089	0.316	0.074
CharNgrams	0.027	0.075	0.009	0.073	0.292	0.057
Periodic Number	0.031	0.128	0.010	0.098	0.361	0.078
BWT	0.027	0.076	0.009	0.073	0.292	0.064
LZ77	0.027	0.077	0.009	0.074	0.301	0.063
PAQ	0.027	0.076	0.009	0.073	0.292	0.060
PPM	0.027	0.077	0.009	0.073	0.293	0.060
Untransformed	0.027	0.080	0.009	0.073	0.296	0.060
Total Average	0.028	0.078	0.009	0.075	0.296	0.059

Table 8. Average Decompression Times (s) for SMILES Data Transform Variations

	SMILES - Average Decompression Times (s)					
Transform Variation	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
Reused Prefixes	0.033	0.119	0.010	0.095	0.334	0.079
Unused Prefixes	0.029	0.097	0.009	0.083	0.299	0.069
Existing_Caps	0.028	0.077	0.009	0.074	0.296	0.059
Existing_Lowercase	0.028	0.078	0.010	0.074	0.296	0.059
Existing_LowerNum	0.028	0.076	0.009	0.074	0.296	0.058
Existing_Numbers	0.027	0.074	0.009	0.072	0.293	0.057
Existing_Unused	0.027	0.072	0.009	0.072	0.288	0.056
Existing_UpperNum	0.027	0.077	0.009	0.074	0.297	0.058
Unused	0.027	0.067	0.008	0.073	0.277	0.055
Reused NumPrefix PeriodicNum	0.033	0.147	0.010	0.107	0.400	0.085
Stars NumPrefix PeriodicNum	0.031	0.130	0.010	0.099	0.367	0.079
Unused NumPrefix PeriodicNum	0.029	0.109	0.009	0.087	0.314	0.071
BWT	0.027	0.076	0.009	0.073	0.292	0.064
LZ77	0.027	0.077	0.009	0.074	0.301	0.063
PAQ	0.027	0.076	0.009	0.073	0.292	0.060
PPM	0.027	0.077	0.009	0.073	0.293	0.060
Untransformed	0.027	0.080	0.009	0.073	0.296	0.060
Total Average	0.028	0.078	0.009	0.075	0.296	0.059

Table 9. Average Decompression Times (s) for SMILES CharNgram Levels

	SMILES - Average Decompression Times (s)					
CharNgram Level	7Zip	BZip2	GZip	PPMd	PPMVC	ZPAQ
CharNgrams_2	0.030	0.086	0.009	0.083	0.310	0.063
CharNgrams_3	0.028	0.073	0.009	0.075	0.295	0.057
CharNgrams_4	0.027	0.070	0.009	0.072	0.291	0.056
CharNgrams_5	0.027	0.071	0.008	0.071	0.289	0.056
CharNgrams_6	0.027	0.071	0.009	0.071	0.289	0.056
CharNgrams_7	0.027	0.073	0.009	0.071	0.288	0.057
CharNgrams_8	0.027	0.074	0.009	0.072	0.288	0.057
CharNgrams_9	0.027	0.075	0.010	0.072	0.288	0.057
CharNgrams_10	0.027	0.077	0.009	0.072	0.288	0.058
Total Average	0.027	0.075	0.009	0.073	0.292	0.057

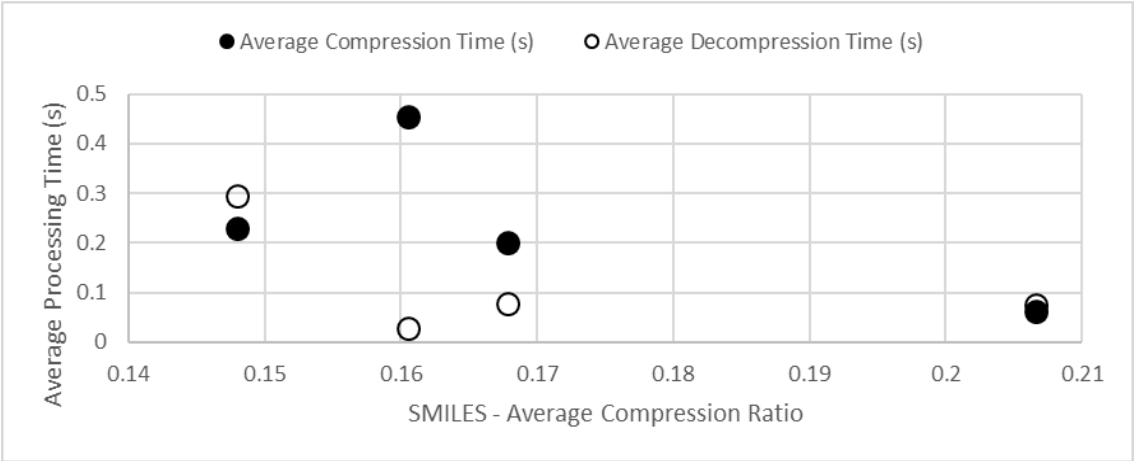

Figure 13. Overall Average Compression and Decompression Times (s) vs Compression Ratios for SMILES Compression Algorithms

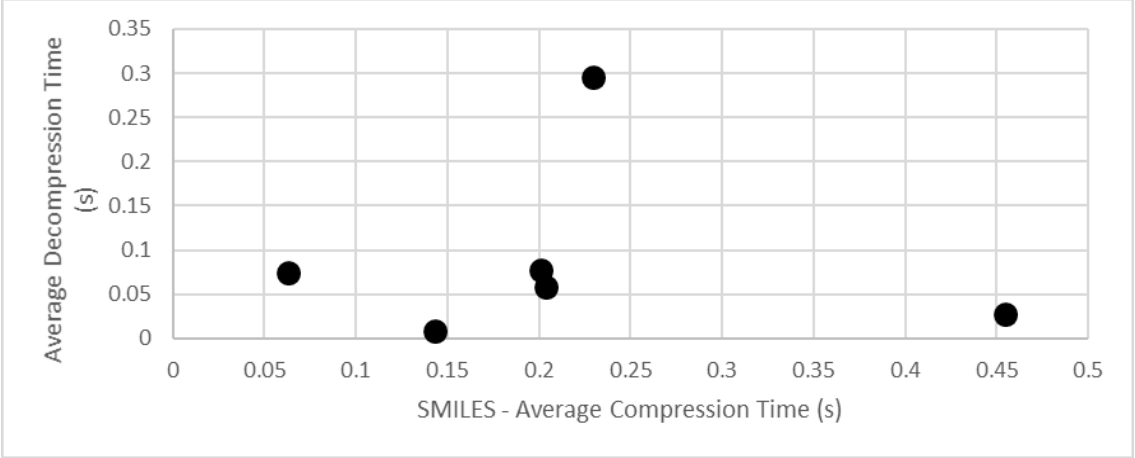

Figure 14. Overall Average Decompression Times (s) vs Compression Times (s) for SMILES Compression Algorithms

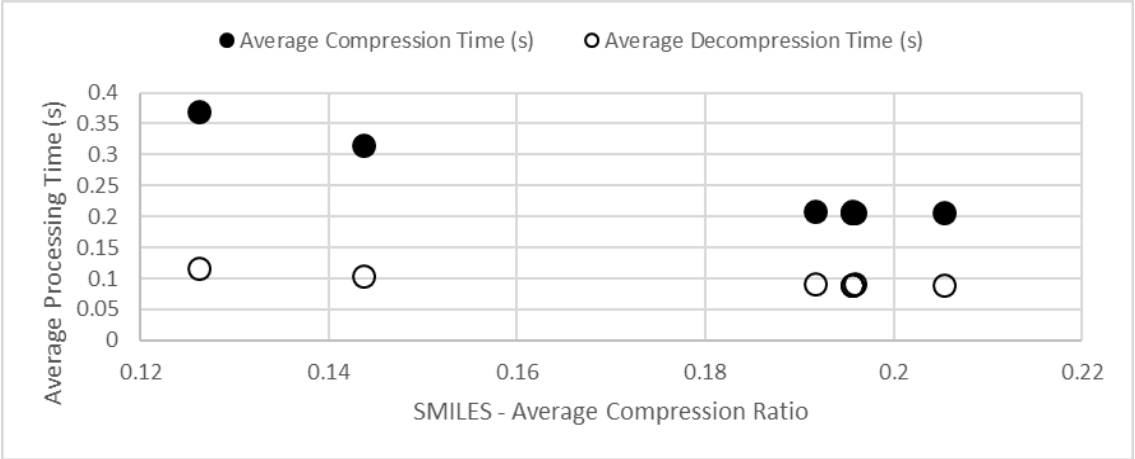

Figure 15. Overall Average Compression and Decompression Times (s) vs Compression Ratios for SMILES Data Transforms

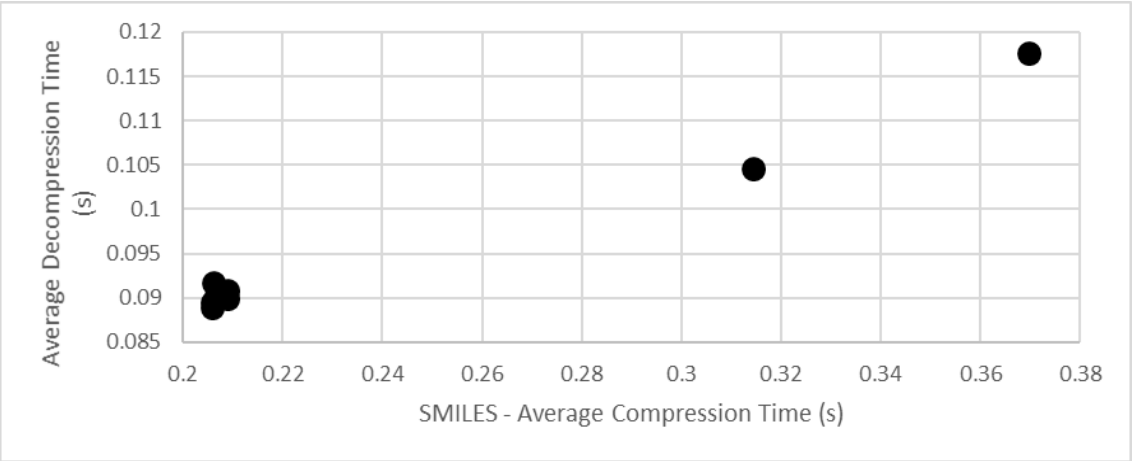

Figure 16. Overall Average Decompression Times (s) vs Compression Times (s) for SMILES Data Transforms

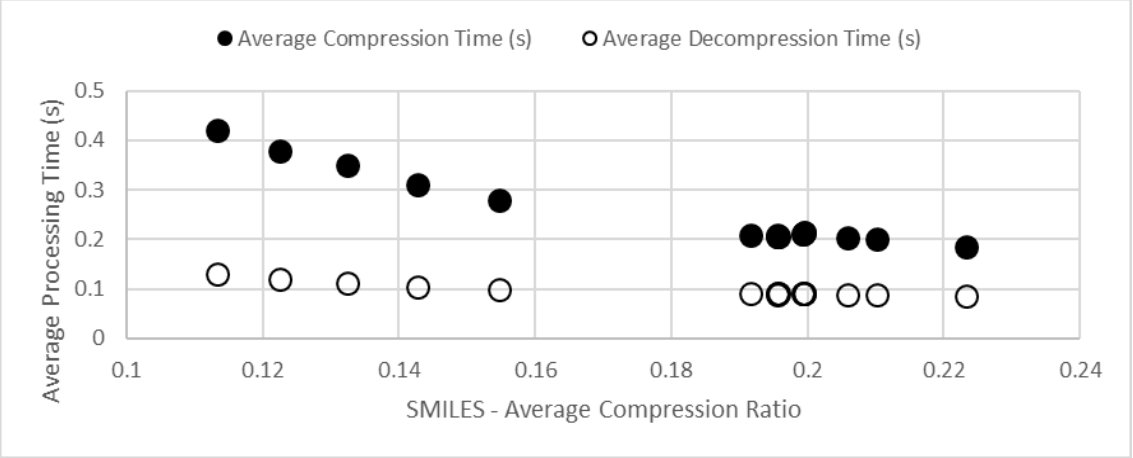

Figure 17. Overall Average Compression and Decompression Times (s) vs Compression Ratios for SMILES Data Transform Variations

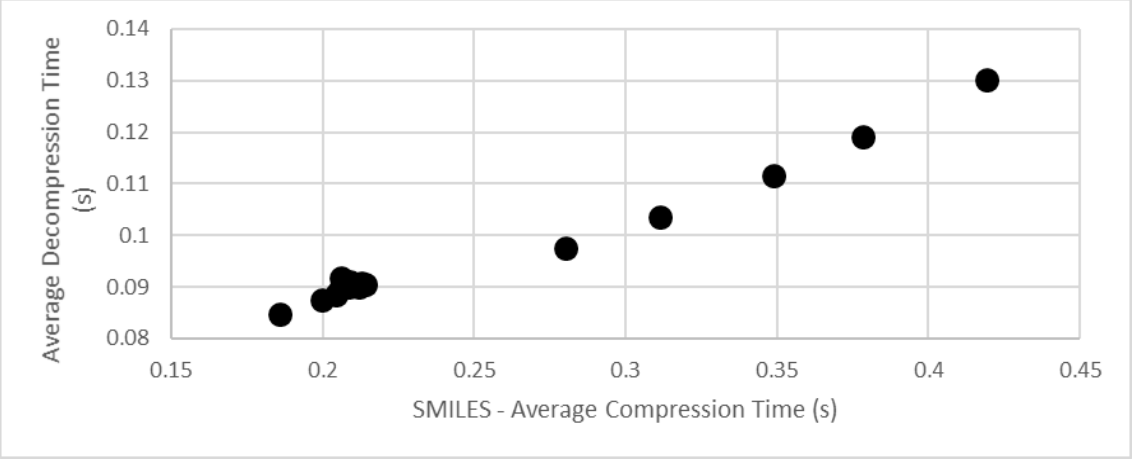

Figure 18. Overall Average Decompression Times (s) vs Compression Times (s) for SMILES Data Transform Variations

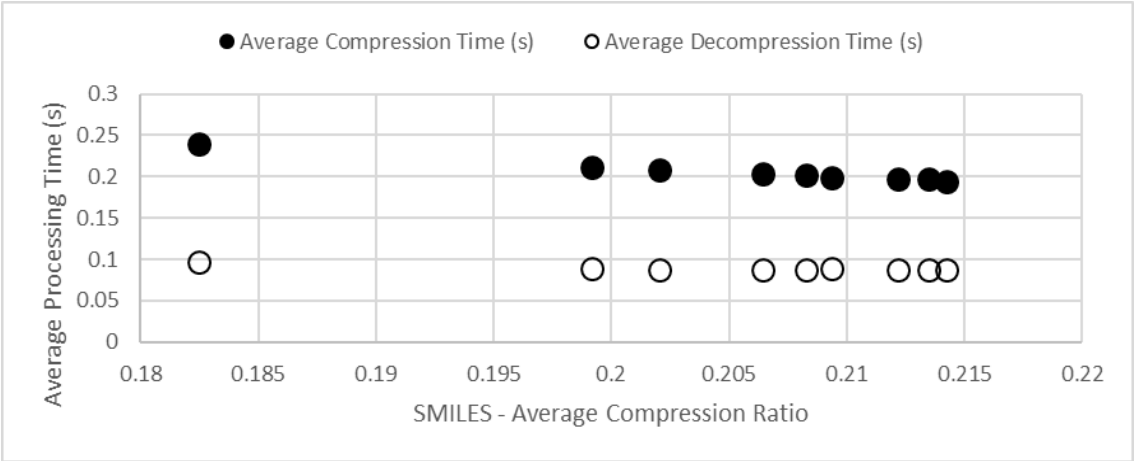

Figure 19. Overall Average Compression and Decompression Times (s) vs Compression Ratios for SMILES CharNgram Levels

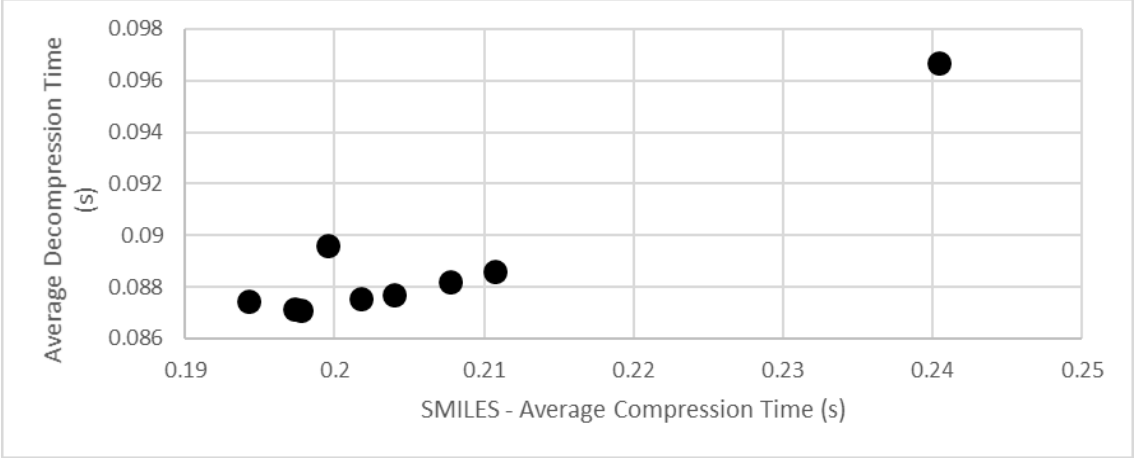

Figure 20. Overall Average Decompression Times (s) vs Compression Times (s) for SMILES CharNgram Levels

Table 10. Average Compression Results for SMILES Compression Algorithm

SMILES Compression Algorithm	Compression Ratio	Compression Time (s)	Decompression Time (s)
7Zip	0.161	0.455	0.028
BZip2	0.168	0.201	0.078
GZip	0.251	0.143	0.009
PPMd	0.207	0.063	0.075
PPMVC	0.148	0.229	0.296
ZPAQ	0.264	0.204	0.059

Table 11. Average Compression Results for SMILES Data Transform

SMILES			
Data Transform	Compression Ratio	Compression Time (s)	Decompression Time (s)
Caps	0.144	0.315	0.105
CharNgrams	0.205	0.206	0.089
Periodic Number	0.126	0.370	0.118
BWT	0.196	0.207	0.090
LZ77	0.196	0.206	0.092
PAQ	0.196	0.206	0.090
PPM	0.196	0.209	0.090
Untransformed	0.192	0.209	0.091

Table 12. Average Compression Results for SMILES Data Transform Variation

SMILES			
Data Transform Variation	Compression Ratio	Compression Time (s)	Decompression Time (s)
Reused Prefixes	0.132	0.349	0.112
Unused Prefixes	0.155	0.280	0.098
Existing_Caps	0.199	0.214	0.090
Existing_Lowercase	0.200	0.213	0.091
Existing_LowerNumber	0.199	0.212	0.090
Existing_Numbers	0.206	0.204	0.089
Existing_Unused	0.210	0.200	0.087
Existing_UpperNumber	0.199	0.212	0.090
Unused	0.223	0.186	0.085
Reused NumPrefix PeriodicNum	0.113	0.419	0.130
Stars NumPrefix PeriodicNum	0.123	0.378	0.119
Unused NumPrefix PeriodicNum	0.143	0.311	0.103
BWT	0.196	0.207	0.090
LZ77	0.196	0.206	0.092
PAQ	0.196	0.206	0.090
PPM	0.196	0.209	0.090
Untransformed	0.192	0.209	0.091

Table 13. Average Compression Results for SMILES CharNgram Level

SMILES CharNgram Level	Compression Ratio	Compression Time (s)	Decompression Time (s)
CharNgrams_2	0.182	0.240	0.097
CharNgrams_3	0.209	0.200	0.090
CharNgrams_4	0.214	0.194	0.087
CharNgrams_5	0.213	0.197	0.087
CharNgrams_6	0.212	0.198	0.087
CharNgrams_7	0.208	0.202	0.088
CharNgrams_8	0.206	0.204	0.088
CharNgrams_9	0.202	0.208	0.088
CharNgrams_10	0.199	0.211	0.089